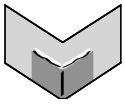
The Journal of Machine Learning Research
Volume 9
Print-Archive Edition

Pages 1437–2880



Microtome Publishing
Brookline, Massachusetts
www.mtome.com

The Journal of Machine Learning Research
Volume 9
Print-Archive Edition

The Journal of Machine Learning Research (JMLR) is an open access journal. All articles published in JMLR are freely available via electronic distribution. This Print-Archive Edition is published annually as a means of archiving the contents of the journal in perpetuity. The contents of this volume are articles published electronically in JMLR in 2008.

JMLR is abstracted in ACM Computing Reviews, INSPEC, and Psychological Abstracts/PsycINFO.

JMLR is a publication of Journal of Machine Learning Research, Inc. For further information regarding JMLR, including open access to articles, visit http://www.jmlr.org/.

JMLR Print-Archive Edition is a publication of Microtome Publishing under agreement with Journal of Machine Learning Research, Inc. For further information regarding the Print-Archive Edition, including subscription and distribution information and background on open-access print archiving, visit Microtome Publishing at http://www.mtome.com/.

ISSN 1532-4435 (print)
ISSN 1533-7928 (online)

JMLR Editorial Board

Journal of Machine Learning Research

Volume 9, 2008

Journal of Machine Learning Research 9 (2008) 1437-1474 Submitted 6/07; Revised 2/08; Published 7/08

Causal Reasoning with Ancestral Graphs

Jiji Zhang JIJI@HSS.CALTECH.EDU
Division of the Humanities and Social Sciences
California Institute of Technology
Pasadena, CA 91106, USA

Editor: Gregory F. Cooper

Abstract

Causal reasoning is primarily concerned with what would happen to a system under external interventions. In particular, we are often interested in predicting the probability distribution of some random variables that would result if some other variables were *forced* to take certain values. One prominent approach to tackling this problem is based on causal Bayesian networks, using directed acyclic graphs as *causal* diagrams to relate post-intervention probabilities to pre-intervention probabilities that are estimable from observational data. However, such causal diagrams are seldom fully testable given observational data. In consequence, many causal discovery algorithms based on data-mining can only output an equivalence class of causal diagrams (rather than a single one). This paper is concerned with causal reasoning given an equivalence class of causal diagrams, represented by a (partial) *ancestral graph*. We present two main results. The first result extends Pearl (1995)'s celebrated *do-calculus* to the context of ancestral graphs. In the second result, we focus on a key component of Pearl's calculus—the property of *invariance under interventions*, and give stronger graphical conditions for this property than those implied by the first result. The second result also improves the earlier, similar results due to Spirtes et al. (1993).

Keywords: ancestral graphs, causal Bayesian network, do-calculus, intervention

1. Introduction

Intellectual curiosity aside, an important reason for people to care about causality or causal explanation is the need—for example, in policy assessment or decision making—to predict consequences of actions or interventions before actually carrying them out. Sometimes we can base that prediction on similar past interventions or experiments, in which case the inference is but an instance of the classical inductive generalization. Other times, however, we do not have access to sufficient controlled experimental studies for various reasons, and can only make passive observations before interventions take place. Under the latter circumstances, we need to reason from pre-intervention or observational data to a post-intervention setting.

A prominent machinery for causal reasoning of this kind is known as *causal Bayesian network* (Spirtes et al., 1993; Pearl, 2000), which we will describe in more detail in the next section. In this framework, once the causal structure—represented by a directed acyclic graph (DAG) over a set of attributes or random variables—is fully given, every query about post-intervention probability can be answered in terms of pre-intervention probabilities. So, if every variable in the causal structure is (passively) observed, the observational data can be used to estimate the post-intervention probability of interest.

Complications come in at least two ways. First, some variables in the causal DAG may be unobserved, or worse, unobservable. So even if the causal DAG (with latent variables) is fully known, we may not be able to predict certain intervention effects because we only have data from the marginal distribution over the observed variables instead of the joint distribution over all causally relevant variables. The question is what post-intervention probability is or is not identifiable given a causal DAG with latent variables. Much of Pearl's work (Pearl, 1995, 1998, 2000), and more recently Tian and Pearl (2004) are paradigmatic attempts to address this problem.

Second, the causal structure is seldom, if ever, fully known. In the situation we are concerned with in this paper, where no substantial background knowledge or controlled study is available, we have to rely upon observational data to inform us about causal structure. The familiar curse is that very rarely can observational data determine a unique causal structure, and many causal discovery algorithms in the literature output an equivalence class of causal structures based on observational data (Spirtes et al., 1993; Meek, 1995a; Spirtes et al., 1999; Chickering, 2002).[1] Different causal structures in the class may or may not give the same answer to a query about post-intervention probability. For a simple illustration, consider two causal Bayesian networks (see Section 2 below), $X \rightarrow Y \rightarrow Z$ and $X \leftarrow Y \rightarrow Z$, over three variables X, Y and Z. The two causal structures are indistinguishable (without strong parametric assumptions) by observational data. Suppose we are interested in the post-intervention probability distribution of Y given that X is manipulated to take some fixed value x. The structure $X \rightarrow Y \rightarrow Z$ entails that the post-intervention distribution of Y is identical to the pre-intervention distribution of Y *conditional on* $X = x$, whereas the structure $X \leftarrow Y \rightarrow Z$ entails that the post-intervention distribution of Y is identical to the pre-intervention marginal distribution of Y. So the two structures give different answers to this particular query. By contrast, if we are interested in the post-intervention distribution of Z under an intervention on Y, the two structures give the same answer.

The matter becomes formidably involved when both complications are present. Suppose we observe a set of random variables $\mathbf{O}$, but for all we know, the underlying causal structure may involve extra latent variables. We will not worry about the estimation of the pre-intervention distribution of $\mathbf{O}$ in this paper, so we may well assume for simplicity that the pre-intervention distribution of $\mathbf{O}$ is known. But we are interested in queries about post-intervention probability, such as the probability of $\mathbf{Y}$ conditional on $\mathbf{Z}$ that would result under an intervention on $\mathbf{X}$ (where $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{O}$). The question is whether and how we can answer such queries from the given pre-intervention distribution of $\mathbf{O}$.

This problem is naturally divided into two parts. The first part is what some causal discovery algorithms attempt to achieve, namely, to learn something about the causal structure—usually features shared by all causal structures in an equivalence class—from the pre-intervention distribution of $\mathbf{O}$. The second part is to figure out, given the learned causal information, whether a post-intervention probability is identifiable in terms of pre-intervention probabilities.

This paper provides some results concerning the second part, assuming the available causal information is summarized in a (partial) *ancestral graph*. Ancestral graphical models (Richardson and Spirtes, 2002, 2003) have proved to be an elegant and useful surrogate for DAG models with latent variables (more details follow in Section 3), not the least because provably correct algorithms are available for learning an equivalence class of ancestral graphs represented by a partial ancestral graph from the pre-intervention distribution of the observed variables—in particular, from the

1. The recent work on linear non-Gaussian structural equation models (Shimizu et al., 2006) is an exception. However, we do not make parametric assumptions in this paper.

conditional independence and dependence relations implied by the distribution (Spirtes et al., 1999; Zhang, forthcoming).

We have two main results. First, we extend the *do*-calculus of Pearl (1995) to the context of ancestral graphs (Section 4), so that the resulting calculus is based on an equivalence class of causal DAGs with latent variables rather than a single one. Second, we focus on a key component of Pearl's calculus—the property of *invariance under interventions* studied by Spirtes et al. (1993), and give stronger graphical conditions for this property than those implied by the first result (Section 5). Our result improves upon the Spirtes-Glymour-Scheines conditions for invariance formulated with respect to the so-called *inducing path graphs*, whose relationship with ancestral graphs is discussed in Appendix A.

2. Causal Bayesian Network

A Bayesian network for a set of random variables $\mathbf{V}$ consists of a pair $\langle \mathcal{G}, P \rangle$, where $\mathcal{G}$ is a directed acyclic graph (DAG) with $\mathbf{V}$ as the set of vertices, and P is the joint probability function of $\mathbf{V}$, such that P factorizes according to $\mathcal{G}$ as follows:

$$P(\mathbf{V}) = \prod_{Y \in \mathbf{V}} P(Y | \mathbf{Pa}_{\mathcal{G}}(Y))$$

where $\mathbf{Pa}_{\mathcal{G}}(Y)$ denotes the set of parents of Y in $\mathcal{G}$. In a causal Bayesian network, the DAG $\mathcal{G}$ is interpreted causally, as a representation of the causal structure over $\mathbf{V}$. That is, for $X, Y \in \mathbf{V}$, an arrow from X to Y ($X \to Y$) in $\mathcal{G}$ means that X has a *direct* causal influence on Y relative to $\mathbf{V}$. We refer to a causally interpreted DAG as a **causal DAG**. The postulate that the (pre-intervention) joint distribution P factorizes according to the causal DAG $\mathcal{G}$ is known as the **causal Markov condition**.

What about interventions? For simplicity, let us focus on what Pearl (2000) calls *atomic* interventions—interventions that fix the values of the target variables—though the results in Section 5 also apply to more general types of interventions (such as interventions that confer a non-degenerate probability distribution on the target variables). In the framework of causal Bayesian network, an intervention on $\mathbf{X} \subseteq \mathbf{V}$ is supposed to be *effective* in the sense that the value of $\mathbf{X}$ is completely determined by the intervention, and *local* in the sense that the conditional distributions of other variables (variables not in $\mathbf{X}$) given their respective parents in the causal DAG are not affected by the intervention. Graphically, such an intervention amounts to erasing all arrows into $\mathbf{X}$ in the causal DAG (because variables in $\mathbf{X}$ do not depend on their original parents any more), but otherwise keeping the graph as it is. Call this modified graph the **post-intervention causal graph**.

Based on this understanding of interventions, the following postulate has been proposed by several authors in various forms (Robins, 1986; Spirtes et al., 1993; Pearl, 2000):

> **Intervention Principle** Given a causal DAG $\mathcal{G}$ over $\mathbf{V}$ and a (pre-intervention) joint distribution P that factorizes according to $\mathcal{G}$, the post-intervention distribution $P_{\mathbf{X}:=\mathbf{x}}(\mathbf{V})$—that is, the joint distribution of $\mathbf{V}$ after $\mathbf{X} \subseteq \mathbf{V}$ are manipulated to values $\mathbf{x}$ by an intervention—takes a similar, truncated form of factorization, as follows:
>
> $$P_{\mathbf{X}:=\mathbf{x}}(\mathbf{V}) = \begin{cases} \prod_{Y \in \mathbf{V} \setminus \mathbf{X}} P(Y | \mathbf{Pa}_{\mathcal{G}}(Y)) & \text{for values of } \mathbf{V} \text{ consistent with } \mathbf{X} = \mathbf{x}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that in the case of a null intervention (when $\mathbf{X} = \emptyset$), the intervention principle implies the factorization of the pre-intervention distribution P according to $\mathcal{G}$, which is just the causal Markov

condition. So the intervention principle generalizes the causal Markov condition: it assumes that the post-intervention distribution also satisfies the causal Markov condition with the post-intervention causal graph.

By the intervention principle, once the causal DAG is given, the post-intervention joint distribution can be calculated in terms of pre-intervention probabilities.[2] So if every variable is observed, and hence those pre-intervention probabilities can be estimated, any post-intervention probability is estimable as well.

It is time to recall the two complications mentioned in the last section. First, the intervention principle is only plausible when the given set of variables is *causally sufficient*. Here is what causal sufficiency means. Given a set of variables $\mathbf{V}$, and two variables $A, B \in \mathbf{V}$, a variable C (not necessarily included in $\mathbf{V}$) is called a *common direct cause* of A and B relative to $\mathbf{V}$ if C has a direct causal influence on A and also a direct causal influence on B relative to $\mathbf{V} \cup \{C\}$. $\mathbf{V}$ is said to be *causally sufficient* if for every pair of variables $V_1, V_2 \in \mathbf{V}$, every common direct cause of V_1 and V_2 relative to $\mathbf{V}$ is also a member of $\mathbf{V}$. It is well known that the causal Markov condition tends to fail for a causally insufficient set of variables (Spirtes et al., 1993), and even more so with the intervention principle. But in most real situations, there is no reason to assume that the set of observed variables is causally sufficient, so the causal Bayesian network may well involve latent variables.

Second, the causal DAG is not fully learnable with observational, pre-intervention data. The causal discovery algorithms in the literature—some of which are provably correct in the large sample limit assuming the causal Markov condition and its converse, causal Faithfulness condition—typically return an equivalence class of DAGs that imply the same conditional independence relations among the observed variables (according to the Markov condition), with some causal features in common that constitute the learned causal information. Given such limited causal information, a post-intervention probability may or may not be uniquely identifiable.

Taking both complications into account, the interesting question is this: what causal reasoning is warranted given the causal information learnable by algorithms that do not assume causal sufficiency for the set of observed variables, such as the FCI algorithm presented in Spirtes et al. (1999)? Before we explore the question, let us make it a little more precise with the formalism of ancestral graphs.

3. Ancestral Graphical Models

Ancestral graphical models are motivated by the need to represent data generating processes that may involve latent confounders and/or selection bias,[3] without explicitly modelling the unobserved variables (Richardson and Spirtes, 2002). We do not deal with selection bias in this paper, so we use only part of the machinery.

A (directed) *mixed graph* is a vertex-edge graph that may contain two kinds of edges: directed edges ($\rightarrow$) and bi-directed edges ($\leftrightarrow$). Between any two vertices there is at most one edge. The two ends of an edge we call *marks*. Obviously there are two kinds of marks: *arrowhead* ($>$) and *tail* ($-$). The marks of a bi-directed edge are both arrowheads, and a directed edge has one arrowhead

2. A technical issue is that some conditional probabilities may be undefined in the pre-intervention distribution. In this paper we ignore that issue by assuming that the pre-intervention distribution is strictly positive. Otherwise we just need to add the proviso "when all the conditional probabilities involved are defined" to all our results.
3. Roughly speaking, there is selection bias if the probability of a unit being sampled depends on certain properties of the unit. The kind of selection bias that is especially troublesome for causal inference is when two or more properties of interest affect the probability of being sampled, giving rise to "misleading" associations in the sample.

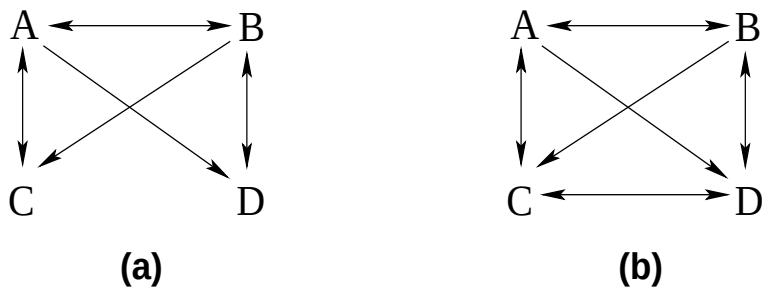


Figure 1: (a) an ancestral graph that is not maximal; (b) a maximal ancestral graph.

and one tail. We say an edge is *into* (or *out of*) a vertex if the mark of the edge at the vertex is an arrowhead (or tail).

Two vertices are said to be *adjacent* in a graph if there is an edge (of any kind) between them. Given a mixed graph $\mathcal{G}$ and two adjacent vertices X, Y therein, X is called a *parent* of Y and Y a *child* of X if $X \rightarrow Y$ is in $\mathcal{G}$; X is called a *spouse* of Y (and Y a spouse of X) if $X \leftrightarrow Y$ is in $\mathcal{G}$. A *path* in $\mathcal{G}$ is a sequence of distinct vertices $\langle V_0, ..., V_n \rangle$ such that for all $0 \leq i \leq n-1$, V_i and V_{i+1} are adjacent in $\mathcal{G}$. A *directed path from V_0 to V_n* in $\mathcal{G}$ is a sequence of distinct vertices $\langle V_0, ..., V_n \rangle$ such that for all $0 \leq i \leq n-1$, V_i is a parent of V_{i+1} in $\mathcal{G}$. X is called an *ancestor* of Y and Y a *descendant* of X if $X = Y$ or there is a directed path from X to Y. We use $\mathbf{Pa}_{\mathcal{G}}, \mathbf{Ch}_{\mathcal{G}}, \mathbf{Sp}_{\mathcal{G}}, \mathbf{An}_{\mathcal{G}}, \mathbf{De}_{\mathcal{G}}$ to denote the set of parents, children, spouses, ancestors, and descendants of a vertex in $\mathcal{G}$, respectively. A *directed cycle* occurs in $\mathcal{G}$ when $Y \rightarrow X$ is in $\mathcal{G}$ and $X \in \mathbf{An}_{\mathcal{G}}(Y)$. An *almost directed cycle* occurs when $Y \leftrightarrow X$ is in $\mathcal{G}$ and $X \in \mathbf{An}_{\mathcal{G}}(Y)$.[4]

Given a path $p = \langle V_0, ..., V_n \rangle$ with $n > 1$, V_i ($1 \leq i \leq n-1$) is a *collider* on p if the two edges incident to V_i are both into V_i, that is, have an arrowhead at V_i; otherwise it is called a *noncollider* on p. In Figure 1(a), for example, B is a collider on the path $\langle A, B, D \rangle$, but is a non-collider on the path $\langle C, B, D \rangle$. A *collider path* is a path on which every vertex except for the endpoints is a collider. For example, in Figure 1(a), the path $\langle C, A, B, D \rangle$ is a collider path because both A and B are colliders on the path. Let $\mathbf{L}$ be any subset of vertices in the graph. An *inducing path relative to* $\mathbf{L}$ is a path on which every vertex not in $\mathbf{L}$ (except for the endpoints) is a collider on the path and every collider is an ancestor of an endpoint of the path. For example, any single-edge path is trivially an inducing path relative to any set of vertices (because the definition does not constrain the endpoints of the path). In Figure 1(a), the path $\langle C, B, D \rangle$ is an inducing path relative to $\{B\}$, but not an inducing path relative to the empty set (because B is not a collider). However, the path $\langle C, A, B, D \rangle$ is an inducing path relative to the empty set, because both A and B are colliders on the path, A is an ancestor of D, and B is an ancestor of C. To simplify terminology, we will henceforth refer to inducing paths relative to the empty set simply as inducing paths.[5]

Definition 1 (MAG) *A mixed graph is called a maximal ancestral graph (MAG) if*

i. the graph does not contain any directed or almost directed cycles (ancestral); and

4. The terminology of "almost directed cycle" is motivated by the fact that removing the arrowhead at Y on $Y \leftrightarrow X$ results in a directed cycle.
5. They are called *primitive inducing paths* by Richardson and Spirtes (2002).

ii. there is no inducing path between any two non-adjacent vertices (maximal).

The first condition is obviously an extension of the defining condition for DAGs. It follows that in an ancestral graph an arrowhead, whether on a directed edge or a bi-directed edge, implies non-ancestorship. The second condition is a technical one, but the original motivation is the familiar pairwise Markov property of DAGs: if two vertices are not adjacent, then they are d-separated by some set of other vertices. The notion of d-separation carries over to mixed graphs in a straightforward way, as we will see shortly. But in general an ancestral graph does not need to satisfy the pairwise Markov property, or what is called maximality here. A sufficient and necessary condition for maximality turns out to be precisely the second clause in the above definition, as proved by Richardson and Spirtes (2002). So although the graph in Figure 1(a) is ancestral, it is not maximal because there is an inducing path between C and D (i.e., $\langle C, A, B, D \rangle$), but C and D are not adjacent. However, each non-maximal ancestral graph has a unique supergraph that is ancestral and maximal. For example, Figure 1(b) is the unique MAG that is also a supergraph of Figure 1(a); the former has an extra bi-directed edge between C and D.

It is worth noting that both conditions in Definition 1 are obviously met by a DAG. Hence, syntactically a DAG is also a MAG, one without bi-directed edges.

An important notion in directed graphical models is that of d-separation, which captures exactly the conditional independence relations entailed by a DAG according to the Markov condition. It is straightforward to extend the notion to mixed graphs, which, following Richardson and Spirtes (2002), we call *m-separation*.

Definition 2 (m-separation) *In a mixed graph, a path p between vertices X and Y is active (or m-connecting) relative to a (possibly empty) set of vertices $\mathbf{Z}$ ($X, Y \notin \mathbf{Z}$) if*

i. every non-collider on p is not a member of $\mathbf{Z}$;

ii. every collider on p is an ancestor of some member of $\mathbf{Z}$.

X and Y are said to be m-separated by $\mathbf{Z}$ if there is no active path between X and Y relative to $\mathbf{Z}$.

Two disjoint sets of variables $\mathbf{X}$ and $\mathbf{Y}$ are m-separated by $\mathbf{Z}$ if every variable in $\mathbf{X}$ is m-separated from every variable in $\mathbf{Y}$ by $\mathbf{Z}$.

In DAGs, obviously, m-separation reduces to d-separation. The (global) Markov property of ancestral graphical models is defined by m-separation.

A nice property of MAGs is that they can represent the marginal independence models of DAGs in the following sense: given any DAG $\mathcal{G}$ over $\mathbf{V} = \mathbf{O} \cup \mathbf{L}$—where $\mathbf{O}$ denotes the set of observed variables, and $\mathbf{L}$ denotes the set of latent variables—there is a MAG over $\mathbf{O}$ alone such that for any disjoint $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{O}$, $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in $\mathcal{G}$ (and hence entailed by $\mathcal{G}$ to be independent conditional on $\mathbf{Z}$) if and only if they are m-separated by $\mathbf{Z}$ in the MAG (and hence entailed by the MAG to be independent conditional on $\mathbf{Z}$). The following construction gives us such a MAG:

Input: a DAG $\mathcal{G}$ over $\langle \mathbf{O}, \mathbf{L} \rangle$
Output: a MAG $\mathcal{M}_{\mathcal{G}}$ over $\mathbf{O}$

1. for each pair of variables $A, B \in \mathbf{O}$, A and B are adjacent in $\mathcal{M}_{\mathcal{G}}$ if and only if there is an inducing path between them relative to $\mathbf{L}$ in $\mathcal{G}$;

2. for each pair of adjacent variables A, B in $\mathcal{M}_\mathcal{G}$, orient the edge as $A \rightarrow B$ in $\mathcal{M}_\mathcal{G}$ if A is an ancestor of B in $\mathcal{G}$; orient it as $A \leftarrow B$ in $\mathcal{M}_\mathcal{G}$ if B is an ancestor of A in $\mathcal{G}$; orient it as $A \leftrightarrow B$ in $\mathcal{M}_\mathcal{G}$ otherwise.

It can be shown that $\mathcal{M}_\mathcal{G}$ is indeed a MAG and represents the marginal independence model over $\mathbf{O}$ (Richardson and Spirtes, 2002; also see Lemma 20 below). More importantly, $\mathcal{M}_\mathcal{G}$ also retains the ancestral relationships—and hence causal relationships under the standard interpretation—among $\mathbf{O}$. So, if $\mathcal{G}$ is the causal DAG for $\langle \mathbf{O}, \mathbf{L} \rangle$, it is fair to call $\mathcal{M}_\mathcal{G}$ the **causal MAG** for $\mathbf{O}$. Henceforth when we speak of a MAG over $\mathbf{O}$ *representing* a DAG over $\langle \mathbf{O}, \mathbf{L} \rangle$, we mean that the MAG is the output of the above construction procedure applied to the DAG.

Different causal DAGs may correspond to the same causal MAG. So essentially a MAG represents a set of DAGs that have the exact same d-separation structures and ancestral relationships among the observed variables. A causal MAG thus carries uncertainty about what the true causal DAG is, but also reveals features that must be satisfied by the underlying causal DAG.

There is then a natural causal interpretation of the edges in MAGs, derivative from the causal interpretation of DAGs. A directed edge from A to B in a MAG means that A is a cause of B (which is a shorthand way of saying that there is a causal pathway from A to B in the underlying DAG); a bi-directed edge between A and B means that A is not a cause of B and B is not a cause of A, which implies that there is a latent common cause of A and B (i.e., there is a latent variable L in the underlying DAG such that there is a directed path from L to A and a directed path from L to B[6]).

We borrow a simple example from Spirtes et al. (1993) to illustrate various concepts and results in this paper. Suppose we are able to observe the following variables: *Income* (I), *Parents' smoking habits* (PSH), *Smoking* (S), *Genotype* (G) and *Lung cancer* (L). The data, for all we know, are generated according to an underlying mechanism which might involve unobserved common causes. Suppose, unknown to us, the structure of the causal mechanism is the one in Figure 2, where *Profession* is an unmeasured common cause of *Income* and *Smoking*.[7]

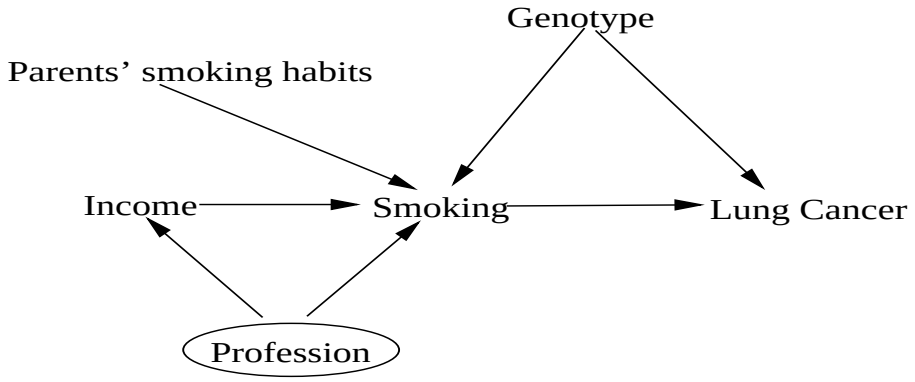


Figure 2: A causal DAG with a latent variable.

6. Note that a latent common cause is not necessarily a common *direct* cause as defined on page 4. The path from L to A, for example, may include other observed variables.
7. This example is used purely for illustrative purposes, so we will not worry why Profession is not observed but Genotype is. The exact domains of the variables do not matter either.

The causal MAG that corresponds to the causal DAG is depicted in Figure 3(a)—which *syntactically* happens to be a DAG in this case. This MAG can represent some other DAGs as well. For example, it can also represent the DAG with an extra latent common cause of *PSH* and *S*.

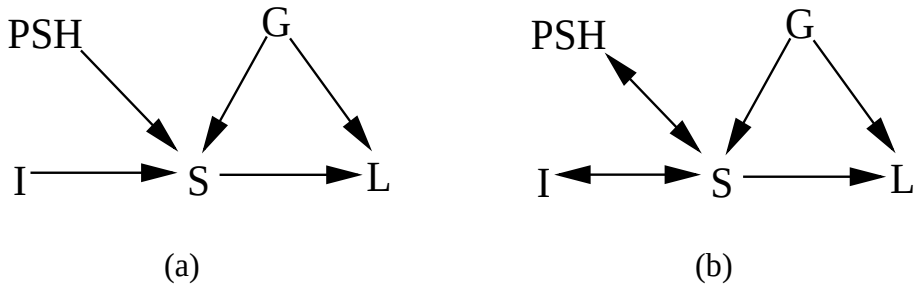


Figure 3: Two Markov Equivalent MAGs.

In general a MAG is still not fully testable with observational data. Just as different DAGs can share the exact same d-separation features and hence entail the exact same conditional independence constraints, different MAGs can entail the exact same constraints by the m-separation criterion. This is known as *Markov equivalence*. Several characterizations of the Markov equivalence between MAGs are available (Spirtes and Richardson, 1996; Ali et al., 2004; Zhang and Spirtes, 2005; Zhao et al., 2005). For the purpose of the present paper, it suffices to note that, as is the case with DAGs, all Markov equivalent MAGs have the same adjacencies and usually some common edge orientations as well. For example, the two MAGs in Figure 3 are Markov equivalent.

This motivates the following representation of equivalence classes of MAGs. Let *partial mixed graphs* denote the class of graphs that can contain four kinds of edges: $\rightarrow$, $\leftrightarrow$, $\circ\!\!-\!\!\circ$ and $\circ\!\!\rightarrow$, and hence three kinds of end marks for edges: arrowhead ($>$), tail ($-$) and circle ($\circ$).

Definition 3 (PAG) *Let $[\mathcal{M}]$ be the Markov equivalence class of an arbitrary MAG $\mathcal{M}$. The partial ancestral graph (PAG) for $[\mathcal{M}]$, $\mathcal{P}_{[\mathcal{M}]}$, is a partial mixed graph such that*

i. $\mathcal{P}_{[\mathcal{M}]}$ has the same adjacencies as $\mathcal{M}$ (and any member of $[\mathcal{M}]$) does;

ii. A mark of arrowhead is in $\mathcal{P}_{[\mathcal{M}]}$ if and only if it is shared by all MAGs in $[\mathcal{M}]$; and

iii. A mark of tail is in $\mathcal{P}_{[\mathcal{M}]}$ if and only if it is shared by all MAGs in $[\mathcal{M}]$.[8]

Basically a PAG represents an equivalence class of MAGs by displaying all common edge marks shared by all members in the class and displaying circles for those marks that are not common, much in the same way that a so-called Pattern (a.k.a. a PDAG or an essential graph) represents an equivalence class of DAGs (see, e.g., Spirtes et al., 1993, chap. 5; Chickering, 1995; Andersson et al., 1997). For instance, the PAG for our running example is drawn in Figure 4, which displays all the commonalities among MAGs that are Markov equivalent to the MAGs in Figure 3.

8. This defines what Zhang (2006, pp. 71) calls *complete* or *maximally oriented* PAGs. In this paper, we do not consider PAGs that fail to display all common edge marks in an equivalence class of MAGs (as, e.g., allowed in Spirtes et al., 1999), so we will simply use 'PAG' to mean 'maximally oriented PAG'.

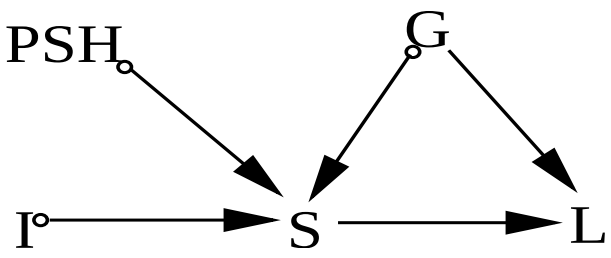


Figure 4: The PAG in our five-variable example.

Different PAGs, representing different equivalence classes of MAGs, entail different sets of conditional independence constraints. Hence a PAG is in principle fully testable by the conditional independence relations among the observed variables. Assuming the causal Markov condition and its converse, the causal Faithfulness condition,[9] there is a provably correct independence-constraint-based algorithm to learn a PAG from an oracle of conditional independence relations (Spirtes et al., 1999; Zhang, 2006, chap. 3).[10] Score-based algorithms for learning PAGs are also under investigation.

Directed paths and ancestors/descendants in a PAG are defined in the same way as in a MAG. In addition, a path between X and Y, $\langle X = V_0, ..., V_n = Y \rangle$, is called a *possibly directed path* from X to Y[11] if for every $0 < i \leq n$, the edge between V_{i-1} and V_i is not into V_{i-1}. Call X a *possible ancestor* of Y (and Y a *possible descendant* of X) if $X = Y$ or there is a possibly directed path from X to Y in the PAG.[12] For example, in Figure 4, the path $\langle I, S, L \rangle$ is a possibly directed path, and I is a possible ancestor of L. We use $\mathbf{PossibleAn}_{\mathcal{P}}(Y)$ to denote the set of possible ancestors of Y in $\mathcal{P}$.

In partial mixed graphs two analogues of m-connecting paths will play a role later. Let p be any path in a partial mixed graph, and W be any (non-endpoint) vertex on p. Let U and V be the two vertices adjacent to W on p. W is a *collider* on p if, as before, both the edge between U and W and the edge between V and W are into W (i.e., have an arrowhead at W, $U *\!\!\rightarrow W \leftarrow\!\!* V$). W is called a *definite non-collider* on p if the edge between U and W or the edge between V and W is out of W

9. We have introduced the causal Markov condition in its factorization form. In terms of d-separation, the causal Markov condition says that d-separation in a causal DAG implies conditional independence in the (pre-intervention) population distribution. The causal Faithfulness condition says that d-connection in a causal DAG implies conditional dependence in the (pre-intervention) population distribution. Given the exact correspondence between d-separation relations among the observed variables in the causal DAG and m-separation relations in the causal MAG, the two conditions imply that conditional independence relations among the observed variables correspond exactly to m-separation in the causal MAG, which forms the basis of constraint-based learning algorithms.
10. It is essentially the FCI algorithm (Spirtes et al., 1999), but with slight modifications (Zhang, 2006, chap. 3). The implemented FCI algorithm in the Tetrad IV package (http://www.phil.cmu.edu/projects/tetrad/tetrad4.html) is the modified version. By the way, if we also take into account the possibility of selection bias, then we need to consider a broader class of MAGs which can contain undirected edges, and the FCI algorithm needs to be augmented with additional edge inference rules (Zhang, 2006, chap. 4; forthcoming).
11. It is named a *potentially directed path* in Zhang (2006, pp. 99). The present terminology is more consistent with the names for other related notions, such as possible ancestor, possibly m-connecting path, etc.
12. The qualifier 'possible/possibly' is used to indicate that there is some MAG represented by the PAG in which the corresponding path is directed, and X is an ancestor of Y. This is not hard to establish given the valid procedure for constructing representative MAGs from a PAG presented in Lemma 4.3.6 of Zhang (2006) or Theorem 2 of Zhang (forthcoming).

(i.e., has a tail at W, $U \leftarrow W *\!\!-\!\!* V$ or $U *\!\!-\!\!* W \rightarrow V$), or both edges have a circle mark at W and there is no edge between U and V (i.e., $U *\!\!-\!\!\circ W \circ\!\!-\!\!* V$, where U and V are not adjacent).[13] The first analogue of m-connecting path is the following:

Definition 4 (Definite m-connecting path) *In a partial mixed graph, a path p between two vertices X and Y is a definite m-connecting path relative to a (possibly empty) set of vertices $\mathbf{Z}$ ($X, Y \notin \mathbf{Z}$) if every non-endpoint vertex on p is either a definite non-collider or a collider and*

i. every definite non-collider on p is not a member of $\mathbf{Z}$;

ii. every collider on p is an ancestor of some member of $\mathbf{Z}$.

It is not hard to see that if there is a definite m-connecting path between X and Y given $\mathbf{Z}$ in a PAG, then in every MAG represented by the PAG, the corresponding path is an m-connecting path between X and Y given $\mathbf{Z}$. For example, in Figure 4 the path $\langle I, S, G\rangle$ is definitely m-connecting given L, and this path is m-connecting given L in every member of the equivalence class. A quite surprising result is that if there is an m-connecting path between X and Y given $\mathbf{Z}$ in a MAG, then there must be a definite m-connecting path (not necessarily the same path) between X and Y given $\mathbf{Z}$ in its PAG, which we will use in Section 5.

Another analogue of m-connecting path is the following:

Definition 5 (Possibly m-connecting path) *In a partial mixed graph, a path p between vertices X and Y is possibly m-connecting relative to a (possibly empty) set of vertices $\mathbf{Z}$ ($X, Y \notin \mathbf{Z}$) if*

i. every definite non-collider on p is not a member of $\mathbf{Z}$;

ii. every collider on p is a possible ancestor of some member of $\mathbf{Z}$.

Obviously a definite m-connecting path is also a possibly m-connecting path, but not necessarily vice versa. In particular, on a possibly m-connecting path it is not required that every (non-endpoint) vertex be of a "definite" status. Figure 5 provides an illustration. The graph on the right is the PAG for the equivalence class that contains the MAG on the left (in this case, unfortunately, no informative edge mark is revealed in the PAG). In the PAG, the path $\langle X, Y, Z, W\rangle$ is a possibly m-connecting path but not a definite m-connecting path relative to $\{Y, Z\}$, because Y and Z are neither colliders nor definite non-colliders on the path. Note that in the MAG, $\langle X, Y, Z, W\rangle$ is not m-connecting relative to $\{Y, Z\}$. In fact, X and W are m-separated by $\{Y, Z\}$ in the MAG. So unlike a definite m-connecting path, a mere possibly m-connecting path in a PAG does not necessarily correspond to a m-connecting path (or imply the existence of a m-connecting path) in a representative MAG in the equivalence class.[14]

As we will see, the main result in Section 4 is formulated in terms of absence of possibly m-connecting paths (what we will call, for want of a better term, definite m-separation), whereas the

13. '*' is used as wildcard that denotes any of the three possible marks: circle, arrowhead, and tail. When the graph is a PAG for some equivalence class of MAGs, the qualifier 'definite' is used to indicate that the vertex is a non-collider on the path in each and every MAG represented by the PAG, even though the circles may correspond to different marks in different MAGs. The reason why $U *\!\!-\!\!\circ W \circ\!\!-\!\!* V$ is a definite non-collider when U and V are not adjacent is because if it were a collider, it would be shared by all Markov equivalent MAGs, and hence would be manifest in the PAG.

14. This case is even more extreme in that in *every* MAG that belongs to the equivalence class, X and W are m-separated by Y and Z. So this example can be used to show that the *do*-calculus developed in Section 4 is not yet complete, though it is not clear how serious the incompleteness is.

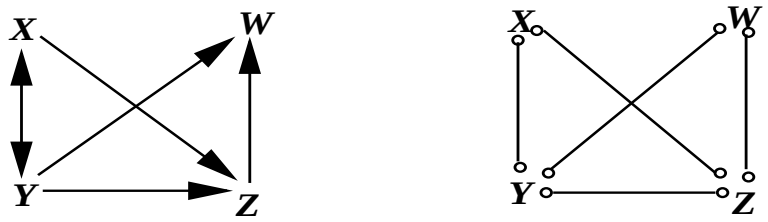


Figure 5: Difference between possible and definite m-connecting paths: in the PAG on the right, $\langle X, Y, Z, W \rangle$ is a possibly m-connecting path relative to $\{Y, Z\}$ but *not* a definite m-connecting path relative to $\{Y, Z\}$. Also note that $\langle X, Y, Z, W \rangle$ is *not* m-connecting relative to $\{Y, Z\}$ in the MAG on the left, even though the MAG is a member of the equivalence class represented by the PAG.

main result in Section 5 is formulated in terms of absence of definite m-connecting paths. This is one important aspect in which the result in Section 5 is better than that in Section 4 (and than the analogous results presented in Spirtes et al., 1993) regarding the property of invariance under interventions. We will come back to this point after we present the PAG-based *do*-calculus.

4. Do-Calculus

Pearl (1995) developed an elegant *do*-calculus for identifying post-intervention probabilities given a single causal DAG with (or without) latent variables. To honor the name of the calculus, in this section we will use Pearl's *'do' operator* to denote post-intervention probabilities. Basically, the notation we used for the post-intervention probability function under an intervention on $\mathbf{X}$, $P_{\mathbf{X}:=\mathbf{x}}(\bullet)$, will be written as $P(\bullet|do(\mathbf{X} = \mathbf{x}))$.

The calculus contains three inference rules whose antecedents make reference to surgeries on the given causal DAG. There are two types of graph manipulations:

Definition 6 (Manipulations of DAGs) *Given a DAG $\mathcal{G}$ and a set of variables $\mathbf{X}$ therein,*

- *the* **X-lower-manipulation** *of $\mathcal{G}$ deletes all edges in $\mathcal{G}$ that are out of variables in $\mathbf{X}$, and otherwise keeps $\mathcal{G}$ as it is. The resulting graph is denoted as $\mathcal{G}_{\underline{\mathbf{X}}}$.*
- *the* **X-upper-manipulation** *of $\mathcal{G}$ deletes all edges in $\mathcal{G}$ that are into variables in $\mathbf{X}$, and otherwise keeps $\mathcal{G}$ as it is. The resulting graph is denoted as $\mathcal{G}_{\overline{\mathbf{X}}}$.*

The following proposition summarizes Pearl's *do*-calculus. (Following Pearl, we use lower case letters to denote generic value settings for the sets of variables denoted by the corresponding upper case letters. So for simplicity we write $P(\mathbf{x})$ to mean $P(\mathbf{X} = \mathbf{x})$, and $do(\mathbf{x})$ to mean $do(\mathbf{X} = \mathbf{x})$.)

Proposition 7 (Pearl) *Let $\mathcal{G}$ be the causal DAG for $\mathbf{V}$, and $\mathbf{U}, \mathbf{X}, \mathbf{Y}, \mathbf{W}$ be disjoint subsets of $\mathbf{V}$. The following rules are sound:*

1. *if $\mathbf{Y}$ and $\mathbf{X}$ are d-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{G}_{\overline{\mathbf{U}}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{w}).$$

2. *if* $\mathbf{Y}$ *and* $\mathbf{X}$ *are d-separated by* $\mathbf{U} \cup \mathbf{W}$ *in* $\mathcal{G}_{\underline{\mathbf{X}}\overline{\mathbf{U}}}$*, then*

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}).$$

3. *if* $\mathbf{Y}$ *and* $\mathbf{X}$ *are d-separated by* $\mathbf{U} \cup \mathbf{W}$ *in* $\mathcal{G}_{\overline{\mathbf{U}\mathbf{X}'}}$*, then*

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{w})$$

where $\mathbf{X}' = \mathbf{X} \backslash \mathbf{An}_{\mathcal{G}_{\overline{\mathbf{U}}}}(\mathbf{W}) = \mathbf{X} \backslash (\cup_{W \in \mathbf{W}} \mathbf{An}_{\mathcal{G}_{\overline{\mathbf{U}}}}(W))$.

The proposition follows from the intervention principle (Pearl, 1995). The first rule is actually not independent—it can be derived from the other two rules (Huang and Valtorta, 2006), but it has long been an official part of the calculus. The soundness of the calculus ensures that any post-intervention probability that can be reduced via the calculus to an expression that only involves pre-intervention probabilities of observed variables is identifiable. Recently, the completeness of the calculus was also established, in the sense that any identifiable post-intervention probability can be so reduced using the calculus (Huang and Valtorta, 2006; Shpister and Pearl, 2006).

Our goal is to develop a similar calculus when the available causal information is given in a PAG. A natural idea is to formulate analogous inference rules in terms of (manipulated) PAGs, to the effect that if a certain rule is applicable given a PAG, the corresponding rule in Pearl's calculus will be applicable given the (unknown) true causal DAG. How to guarantee that? Recall that a PAG represents an equivalence class of MAGs; each MAG, in turn, represents a set of causal DAGs. The union of all these sets is the set of DAGs represented by the PAG—one of them is the true causal DAG. So a sure way to get what we want is to formulate analogous rules in terms of PAGs such that if the rule is applicable given a PAG, then for every DAG represented by the PAG, the corresponding rule in Pearl's calculus is applicable.

For this purpose, it is natural to develop the desired calculus in two steps. First, we derive an analogous *do*-calculus based on MAGs, such that if a rule is applicable given a MAG, then for every DAG represented by the MAG, the corresponding rule in Pearl's calculus is applicable. Second, we extend that to a *do*-calculus based on PAGs, such that if a rule is applicable given a PAG, then for every MAG in the equivalence class represented by the PAG, the corresponding rule in the MAG-based calculus is applicable.

Before we define appropriate analogues of graph manipulations on MAGs, it is necessary to distinguish two kinds of directed edges in a MAG, according to the following criterion.

Definition 8 (Visibility) *Given a MAG* $\mathcal{M}$*, a directed edge* $A \to B$ *in* $\mathcal{M}$ *is* **visible** *if there is a vertex* C *not adjacent to* B*, such that either there is an edge between* C *and* A *that is into* A*, or there is a collider path between* C *and* A *that is into* A *and every vertex on the path is a parent of* B*. Otherwise* $A \to B$ *is said to be* **invisible**.

Figure 6 gives the possible configurations that make a directed edge $A \to B$ visible. The distinction between visible and invisible directed edges is important because of the following two facts.

Lemma 9 *Let* $\mathcal{G}$ *be a DAG over* $\mathbf{O} \cup \mathbf{L}$*, and* $\mathcal{M}$ *be the MAG over* $\mathbf{O}$ *that represents the DAG. For any* $A, B \in \mathbf{O}$*, if* $A \in \mathbf{An}_{\mathcal{G}}(B)$*, and there is an inducing path relative to* $\mathbf{L}$ *between* A *and* B *that is into* A *in* $\mathcal{G}$*, then there is a directed edge* $A \to B$ *in* $\mathcal{M}$ *that is invisible.*

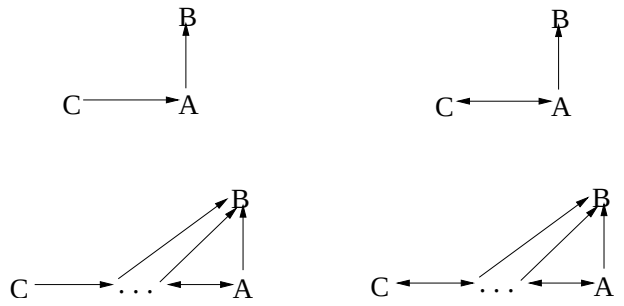


Figure 6: Possible configurations of visibility for $A \rightarrow B$.

Proof See Appendix B. ■

Taking the contrapositive of Lemma 9 gives us the fact that if $A \rightarrow B$ is visible in a MAG, then in *every* DAG represented by the MAG, there is no inducing path between A and B relative to the set of latent variables that is also into A. This implies that for every such DAG G, $G_{\underline{A}}$—the graph resulting from eliminating edges out of A in G—will not contain any inducing path between A and B relative to the set of latent variables, which means that the MAG that represents $G_{\underline{A}}$ will not contain any edge between A and B. So intuitively, deleting edges out of A in the underlying DAG corresponds to deleting visible arrows out of A in the MAG.

How about invisible arrows? Here is the relevant fact.

Lemma 10 *Let $\mathcal{M}$ be any MAG over a set of variables* **O**, *and $A \rightarrow B$ be any directed edge in $\mathcal{M}$. If $A \rightarrow B$ is invisible in $\mathcal{M}$, then there is a DAG whose MAG is $\mathcal{M}$ in which A and B share a latent parent, that is, there is a latent variable L_{AB} in the DAG such that $A \leftarrow L_{AB} \rightarrow B$ is a subgraph of the DAG.*

Proof See Appendix B. ■

Obviously $A \leftarrow L_{AB} \rightarrow B$ is an inducing path between A and B relative to the set of latent variables. So if $A \rightarrow B$ in a MAG is invisible, at least for *some* DAG G represented by the MAG—and for all we know, this DAG may well be the true causal DAG—$G_{\underline{A}}$ contains $A \leftarrow L_{AB} \rightarrow B$, and hence corresponds to a MAG in which $A \leftrightarrow B$ appears.

Finally, for either $A \leftrightarrow B$ or $A \rightarrow B$ in a MAG, it is not hard to show that for *every* DAG represented by the MAG, there is no inducing path in the DAG between A and B relative to the set of latent variables that is also out of B (since otherwise B would be an ancestor of A, violating the definition of ancestral graphs). So deleting edges into B in the underlying DAG corresponds to deleting edges into B in the MAG. These considerations motivate the following definition.

Definition 11 (Manipulations of MAGs) *Given a MAG $\mathcal{M}$ and a set of variables* **X** *therein,*

- *the* **X-lower-manipulation** *of $\mathcal{M}$ deletes all those edges that are visible in $\mathcal{M}$ and are out of variables in* **X**, *replaces all those edges that are out of variables in* **X** *but are invisible in $\mathcal{M}$ with bi-directed edges, and otherwise keeps $\mathcal{M}$ as it is. The resulting graph is denoted as $\mathcal{M}_{\underline{\mathbf{X}}}$.*

- *the* **X-upper-manipulation** *of $\mathcal{M}$ deletes all those edges in $\mathcal{M}$ that are into variables in* **X**, *and otherwise keeps $\mathcal{M}$ as it is. The resulting graph is denoted as $\mathcal{M}_{\overline{\mathbf{X}}}$.*

We stipulate that lower-manipulation has a higher priority than upper-manipulation, so that $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ (or $\mathcal{M}_{\overline{\mathbf{X}}\underline{\mathbf{Y}}}$) denotes the graph resulting from applying the **X***-upper-manipulation to the* **Y***-lower-manipulated graph of $\mathcal{M}$.*

A couple of comments are in order. First, unlike the case of DAGs, the lower-manipulation for MAGs may introduce new edges, that is, replacing invisible directed edges with bi-directed edges. Again, the reason we do this is that an invisible directed edge from A to B allows the possibility of a latent common parent of A and B in the underlying DAG. If so, the A-lower-manipulated DAG will correspond to a MAG in which there is a bi-directed edge between A and B. Second, because of the possibility of introducing new bi-directed edges, we need the priority stipulation that lower-manipulation is to be done before upper-manipulation. The stipulation is not necessary for DAGs, because no new edges would be introduced in the lower-manipulation of DAGs, and hence the order does not matter.

Ideally, if $\mathcal{M}$ is the MAG of a DAG $\mathcal{G}$, we would like $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ to be the MAG of $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. But this is not always possible, as two DAGs represented by the same MAG before a manipulation may correspond to different MAGs after the manipulation. But we still have the following fact:

Lemma 12 *Let $\mathcal{G}$ be a DAG over $\mathbf{O} \cup \mathbf{L}$, and $\mathcal{M}$ be the MAG of $\mathcal{G}$ over* **O**. *Let* **X** *and* **Y** *be two possibly empty subsets of* **O**, *and $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$ be the MAG of $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. For any $A, B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if there is an m-connecting path between A and B given* **C** *in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, then there is an m-connecting path between A and B given* **C** *in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$.*

Proof See Appendix B. ■

Recall that a graphical model is called an *independence map* of another if any independence implied by the former is also implied by the latter (Chickering, 2002). So another way of putting Lemma 12 is that $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is an independence map of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, which we write as $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}} \leq \mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. The diagram in Figure 7 visualizes what is going on.

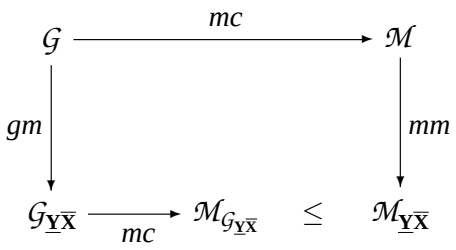


Figure 7: Illustration of Lemma 12: *mc* refers to MAG construction introduced in Section 3; *gm* refers to DAG manipulation; and *mm* refers to MAG manipulation.

Corollary 13 *Let $\mathcal{M}$ be a MAG over* **O**, *and* **X** *and* **Y** *be two subsets of* **O**. *For any $A, B \in$* **O** *and* **C** $\subseteq$ **O** *that does not contain A or B, if A and B are m-separated by* **C** *in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, then A and B are d-separated by* **C** *in $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ for every $\mathcal{G}$ represented by $\mathcal{M}$.*

Proof By Lemma 12, if A and B are m-separated by **C** in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, they are also m-separated by **C** in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, for every $\mathcal{G}$ represented by $\mathcal{M}$, which in turn implies that A and B are d-separated by **C** in $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ for every $\mathcal{G}$ represented by $\mathcal{M}$, because d-separation relations among **O** in a DAG correspond exactly to m-separation relations in its MAG. ■

The converse of Corollary 13, however, is not true in general. To give the simplest example, consider the MAG $\mathcal{M}$ in Figure 8(a): $X \leftarrow Y \rightarrow Z$ (which happens to be a DAG syntactically). The two DAGs, $\mathcal{G}1$ in 8(b) and $\mathcal{G}2$ in 8(c), are both represented by $\mathcal{M}$. By the definition of lower-manipulation, $\mathcal{M}_{\underline{Y}}$ is the graph $X \leftrightarrow Y \leftrightarrow Z$. On the other hand, $\mathcal{G}1_{\underline{Y}}$ is $X \leftarrow L1 \rightarrow Y \quad Z$; and $\mathcal{G}2_{\underline{Y}}$ is $X \quad Y \leftarrow L2 \rightarrow Z$. Obviously, the MAG of $\mathcal{G}1_{\underline{Y}}$ is $X \leftrightarrow Y \quad Z$, and the MAG of $\mathcal{G}2_{\underline{Y}}$ is $X \quad Y \leftrightarrow Z$, both of which are *proper* subgraphs of $\mathcal{M}_{\underline{Y}}$. So an m-separation relation in $\mathcal{M}_{\underline{Y}}$—for example, X and Z are m-separated by the empty set—corresponds to a d-separation relation in both $\mathcal{G}1_{\underline{Y}}$ and $\mathcal{G}2_{\underline{Y}}$, in accord with Corollary 13.

By contrast, the converse of Corollary 13 fails for $\mathcal{M}$. It can be checked that for every $\mathcal{G}$ represented by $\mathcal{M}$, X and Z are d-separated by Y in $\mathcal{G}_{\underline{Y}}$, as evidenced by $\mathcal{G}1_{\underline{Y}}$ and $\mathcal{G}2_{\underline{Y}}$. But X and Z are not m-separated by Y in $\mathcal{M}_{\underline{Y}}$.

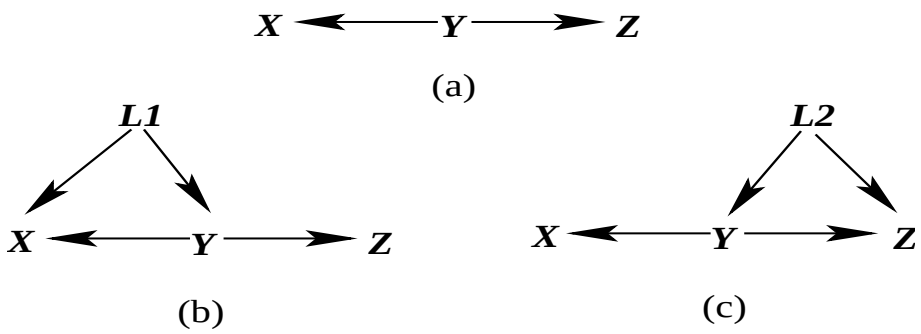


Figure 8: A counterexample to the converse of Corollary 13.

However, Definition 11 is not to be blamed for this limitation. In this simple example, one can easily enumerate all possible directed mixed graphs over X, Y, Z and see that for none of them do both Corollary 13 and its converse hold. Intuitively, this is because the MAG in Figure 8(a) implies that either $\langle X, Y \rangle$ does not have a common latent parent or $\langle Y, Z \rangle$ does not have a common latent parent in the underlying DAG. So under the Y-lower-manipulation of the underlying DAG, for all we know, either $\langle X, Y \rangle$ or $\langle Y, Z \rangle$ will become unconnected. But this disjunctive information cannot be precisely represented by a single graph.

More generally, no matter how we define $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, as long as it is a single graph, the converse of Corollary 13 will not hold in general, unless Corollary 13 itself fails. $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, as a single graph, can only aim to be a supergraph (up to Markov equivalence) of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$ for every $\mathcal{G}$ represented by $\mathcal{M}$ (which makes Corollary 13 true). To this end, Definition 11 is 'minimal' in the following sense: two

variables are adjacent in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ if and only if there exists a DAG $\mathcal{G}$ represented by $\mathcal{M}$ such that the two variables are adjacent in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$. In this regard, $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ does not have more edges than necessary. One can, for example, check this fact for the simple case in Figure 8.

We are now ready to state the intermediate theorem on MAG-based *do*-calculus.

Theorem 14 (*do*-calculus given a MAG) *Let $\mathcal{M}$ be the causal MAG over $\mathbf{O}$, and $\mathbf{U}, \mathbf{X}$, $\mathbf{Y}$, $\mathbf{W}$ be disjoint subsets of $\mathbf{O}$. The following rules are valid, in the sense that if the antecedent of the rule holds, then the consequent holds no matter which DAG represented by $\mathcal{M}$ is the true causal DAG.*

1. *if $\mathbf{Y}$ and $\mathbf{X}$ are m-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{M}_{\overline{\mathbf{U}}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{w}).$$

2. *if $\mathbf{Y}$ and $\mathbf{X}$ are m-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{M}_{\underline{\mathbf{X}}\overline{\mathbf{U}}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}).$$

3. *if $\mathbf{Y}$ and $\mathbf{X}$ are m-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{M}_{\overline{\mathbf{U}\mathbf{X}'}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{w})$$

where $\mathbf{X}' = \mathbf{X} \backslash \mathbf{An}_{\mathcal{M}_{\overline{\mathbf{U}}}}(\mathbf{W})$.

Proof This readily follows from Proposition 7, Corollary 13, and the fact that for every $\mathcal{G}$ represented by $\mathcal{M}$, $\mathbf{An}_{\mathcal{G}_{\overline{\mathbf{U}}}}(\mathbf{W}) \cap \mathbf{O} = \mathbf{An}_{\mathcal{M}_{\overline{\mathbf{U}}}}(\mathbf{W})$. ■

As already noted, the true causal MAG is not uniquely recoverable from the pre-intervention distribution, thanks to Markov equivalence. So the main value of Theorem 14 is to facilitate the development of a PAG-based *do*-calculus. However, it is worth noting that when supplemented with some background causal knowledge, such as knowledge of the form that some variable is not a cause of another variable, it is in principle possible to determine that the true causal MAG belongs to a proper subset of the full equivalence class represented by the PAG. Depending on how strong the background knowledge is, the subset could be as big as the full equivalence class or as small as a singleton. In this sense, Theorem 14 and Theorem 17 below may be viewed as two extreme cases of a more general *do*-calculus based on a subset of Markov equivalent MAGs.

To extend the calculus to PAGs, we need to define manipulations on PAGs. They are essentially the same as the manipulations of MAGs. The definition of visibility still makes sense in PAGs, except that we will call a directed edge in a PAG *definitely visible* if it satisfies the condition for visibility in Definition 8, in order to emphasize that this edge is visible in all MAGs in the equivalence class. Despite the extreme similarity to manipulations on MAGs, let us still write down the definition of PAG manipulations for easy reference.

Definition 15 (Manipulations of PAGs) *Given a PAG $\mathcal{P}$ and a set of variables $\mathbf{X}$ therein,*

- *the* **X-lower-manipulation** *of $\mathcal{P}$ deletes all those edges that are definitely visible in $\mathcal{P}$ and are out of variables in $\mathbf{X}$, replaces all those edges that are out of variables in $\mathbf{X}$ but are not definitely visible in $\mathcal{P}$ with bi-directed edges, and otherwise keeps $\mathcal{P}$ as it is. The resulting graph is denoted as $\mathcal{P}_{\underline{\mathbf{X}}}$.*

- *the* **X-upper-manipulation** *of $\mathcal{P}$ deletes all those edges in $\mathcal{P}$ that are into variables in* **X**, *and otherwise keeps $\mathcal{P}$ as it is. The resulting graph is denoted as $\mathcal{P}_{\overline{\mathbf{X}}}$.*

We stipulate that lower-manipulation has a higher priority than upper-manipulation, so that $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ (or $\mathcal{P}_{\overline{\mathbf{X}}\underline{\mathbf{Y}}}$) denotes the graph resulting from applying the **X***-upper-manipulation to the* **Y***-lower-manipulated graph of $\mathcal{P}$.*

We should emphasize that except in rare situations, $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is not a PAG any more (i.e., not a PAG for any Markov equivalence class of MAGs). But from $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ we still gain information about m-separation in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, where $\mathcal{M}$ is a MAG that belongs to the Markov equivalence class represented by $\mathcal{P}$. Here is a simple connection. Given a MAG $\mathcal{M}$ and the PAG $\mathcal{P}$ that represents $[\mathcal{M}]$, a trivial fact is that a m-connecting path in $\mathcal{M}$ is also a possibly m-connecting path in $\mathcal{P}$. This is also true for $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ and $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$.

Lemma 16 *Let $\mathcal{M}$ be a MAG over* **O**, *and $\mathcal{P}$ be the PAG for $[\mathcal{M}]$. Let* **X** *and* **Y** *be two subsets of* **O**. *For any $A, B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if a path p between A and B is m-connecting given* **C** *in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, then p, the same sequence of variables, forms a possibly m-connecting path between A and B given* **C** *in $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$.*[15]

Proof See Appendix B. ■

If there is no possibly m-connecting path between *A* and *B* given **C** in a partial mixed graph, we say that *A* and *B* are *definitely m-separated* by **C** in the graph. A *do*-calculus follows:

Theorem 17 (*do*-calculus given a PAG) *Let $\mathcal{P}$ be the causal PAG for* **O**, *and* $\mathbf{U}, \mathbf{X}, \mathbf{Y}, \mathbf{W}$ *be disjoint subsets of* **O**. *The following rules are valid:*

1. *if* **Y** *and* **X** *are definitely m-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{P}_{\overline{\mathbf{U}}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{w}).$$

2. *if* **Y** *and* **X** *are definitely m-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{P}_{\underline{\mathbf{X}}\overline{\mathbf{U}}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}).$$

3. *if* **Y** *and* **X** *are definitely m-separated by $\mathbf{U} \cup \mathbf{W}$ in $\mathcal{P}_{\overline{\mathbf{U}\mathbf{X}'}}$, then*

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{w})$$

where $\mathbf{X}' = \mathbf{X} \backslash \mathbf{PossibleAn}_{\mathcal{P}_{\overline{\mathbf{U}}}}(\mathbf{W})$.

15. For our purpose, what we need is the obvious consequence of the lemma that if there is an m-connecting path in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, then there is a possibly m-connecting path in $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. We suspect that a stronger result might hold as well: if there is an m-connecting path in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, then there is a definite m-connecting path in $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. We can't prove or disprove the stronger result at the moment.

Proof It follows from Lemma 16 and Theorem 14. The only caveat is that in general $\mathbf{An}_{\mathcal{M}_{\overline{\mathbf{U}}}}(\mathbf{W}) \neq \mathbf{PossibleAn}_{\mathcal{P}_{\overline{\mathbf{U}}}}(\mathbf{W})$ for an arbitrary $\mathcal{M}$ represented by $\mathcal{P}$. But it is always the case that $\mathbf{An}_{\mathcal{M}_{\overline{\mathbf{U}}}}(\mathbf{W}) \subseteq \mathbf{PossibleAn}_{\mathcal{P}_{\overline{\mathbf{U}}}}(\mathbf{W})$, which means that $\mathbf{X}\backslash\mathbf{An}_{\mathcal{M}_{\overline{\mathbf{U}}}}(\mathbf{W}) \supseteq \mathbf{X}\backslash\mathbf{PossibleAn}_{\mathcal{P}_{\overline{\mathbf{U}}}}(\mathbf{W})$ for every $\mathcal{M}$ represented by $\mathcal{P}$. So it is possible that for rule (3), $\mathcal{P}_{\overline{\mathbf{UX}'}}$ leaves more edges in than necessary, but it does not affect the validity of rule (3). ■

The possibility that $\mathcal{P}_{\overline{\mathbf{UX}'}}$ leaves more edges in than necessary is one of three aspects in which our *do*-calculus may be "incomplete" in the following sense: it is possible that a rule in the PAG-based *do*-calculus is not applicable, but for every DAG compatible with the given PAG, the corresponding rule in Pearl's DAG-based calculus is applicable. The other two aspects are already noted: (1) the calculus is formulated in terms of the absence of possibly m-connecting paths (cf. Footnote 14, and more on this in the next section); and (2) the MAG-based *do*-calculus is based on Corollary 13 whose converse does not hold. Therefore, the PAG-based *do*-calculus as currently formulated may be further improved.

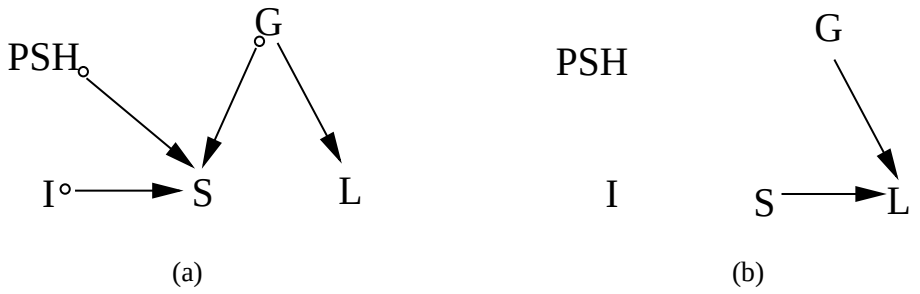


Figure 9: PAG Surgery: $\mathcal{P}_{\underline{S}}$ and $\mathcal{P}_{\overline{S}}$.

That said, let us illustrate the utility of the *do*-calculus with the simple example used in Section 3. Given the PAG in Figure 4 we can infer that $P(L|do(S),G) = P(L|S,G)$ by rule 2, because L and S are definitely m-separated by $\{G\}$ in $\mathcal{P}_{\underline{S}}$ (Figure 9(a)); and $P(G|do(S)) = P(G)$ by rule 3, because G and S are definitely m-separated in $\mathcal{P}_{\overline{S}}$ (Figure 9(b)). It follows that

$$\begin{aligned} P(L|do(S)) &= \sum_G P(L,G|do(S)) \\ &= \sum_G P(L|do(S),G)P(G|do(S)) \\ &= \sum_G P(L|S,G)P(G). \end{aligned}$$

By contrast, it is not valid in the *do*-calculus that $P(L|do(G),S) = P(L|G,S)$ because L and G are not definitely m-separated by $\{S\}$ in $\mathcal{P}_{\underline{G}}$, which is depicted in Figure 10. (Notice the bi-directed edge between L and G.)

5. Invariance Under Interventions

We now develop stronger results for a key component of *do*-calculus, the property of *invariance under interventions*, first systematically studied in Spirtes et al. (1993). The idea is simple. A

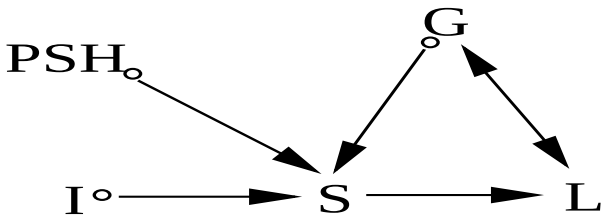


Figure 10: PAG Surgery: $\mathcal{P}_{\underline{G}}$.

conditional probability $P(\mathbf{Y} = \mathbf{y}|\mathbf{Z} = \mathbf{z})$ is said to be *invariant* under an intervention $\mathbf{X} := \mathbf{x}$—or $do(\mathbf{X} = \mathbf{x})$—if $P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{z}) = P(\mathbf{y}|\mathbf{z})$.[16] This concept (under the name of 'observability') plays an important role in some interesting theoretical work on observational studies (e.g., Pratt and Schlaifer, 1988; for a good review see Winship and Morgan, 1999), and also forms the basis of the prediction algorithm presented in Spirtes et al. (1993), which seeks to identify a post-intervention probability by searching for an expression in terms of invariant probabilities.

It is also the corner stone of Pearl's *do*-calculus. To see this, let us take a closer look at the second and third rules in the *do*-calculus. The second rule of the calculus gives a graphical condition for when we can conclude

$$P(\mathbf{y}|do(\mathbf{u}), do(\mathbf{x}), \mathbf{w}) = P(\mathbf{y}|do(\mathbf{u}), \mathbf{x}, \mathbf{w}).$$

If we take $\mathbf{U}$ to be the empty set and write the above equation in the subscript notation, we get

$$P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{w}) = P(\mathbf{y}|\mathbf{x}, \mathbf{w}).$$

Since $P_{\mathbf{X}:=\mathbf{x}}(\mathbf{X} = \mathbf{x}) = 1$, thanks to the supposed effectiveness of the intervention, we have

$$P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{x}, \mathbf{w}).$$

So a special case of the second rule is a condition for $P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = P(\mathbf{y}|\mathbf{x}, \mathbf{w})$, that is, for when $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$ is invariant under the intervention $\mathbf{X} := \mathbf{x}$. In fact, the second rule is nothing but a generalization of this condition to tell when a post-intervention probability $P_{\mathbf{u}}(\mathbf{y}|\mathbf{x}, \mathbf{w})$ would be invariant under a *further* intervention $\mathbf{X} := \mathbf{x}$.

The third rule is more obviously about invariance. It is a generalization of the condition for $P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{w}) = P(\mathbf{y}|\mathbf{w})$, that is, for when $P(\mathbf{y}|\mathbf{w})$ is invariant under the intervention $\mathbf{X} := \mathbf{x}$. The difference between rule 2 and rule 3 is that rule 2 is about invariance of $P(\mathbf{y}|\mathbf{z})$ under an intervention on $\mathbf{X}$ in case $\mathbf{X} \subseteq \mathbf{Z}$ $(= \mathbf{X} \cup \mathbf{W})$, whereas rule 3 is about invariance of $P(\mathbf{y}|\mathbf{z})$ under an intervention on $\mathbf{X}$ in case $\mathbf{X}$ and $\mathbf{Z}$ $(= \mathbf{W})$ are disjoint. As we mentioned earlier, the first rule is not essential, so the *do*-calculus is in effect a generalization of conditions for invariance.

We now focus on this key component of *do*-calculus, and present better graphical conditions for judging invariance given a PAG than those that are implied by the PAG-based *do*-calculus presented in the last section. The conditions for invariance implied by Pearl's (DAG-based) *do*-calculus can

16. Here we allow that $\mathbf{X}$ and $\mathbf{Z}$ have a non-empty intersection, and assume that the conditioning operation is applied to the post-intervention population (i.e., intervening comes before conditioning). As a result, when we speak of $P_{\mathbf{X}:=\mathbf{x}}(\mathbf{y}|\mathbf{z})$, we implicitly assume that $\mathbf{x}$ and $\mathbf{z}$ are consistent regarding the values for variables in $\mathbf{X} \cap \mathbf{Z}$, for otherwise the quantity is undefined.

be equivalently formulated without referring to manipulated graphs, as given in Spirtes et al. (1993, Theorem 7.1) before the *do*-calculus was invented. In this section we develop corresponding conditions in terms of PAGs. The conditions will be not only sufficient in the sense that if the conditions are satisfied, then every DAG compatible with the given PAG entails invariance, but also necessary in the sense that if the conditions fail, then there is at least one DAG compatible with the given PAG that does not entail invariance. In this aspect, the conditions are also superior to earlier results on invariance given an equivalence class of DAGs due to Spirtes et al. (1993, Theorems 7.3 and 7.4).

We first state the conditions for judging invariance given a DAG, originally presented in Spirtes et al. (1993, Theorem 7.1).

Proposition 18 (Spirtes, Glymour, Scheines) *Let $\mathcal{G}$ be the causal DAG for $\mathbf{O}\cup\mathbf{L}$, and $\mathbf{X},\mathbf{Y},\mathbf{Z}\subseteq\mathbf{O}$ be three sets of variables such that $\mathbf{X}\cap\mathbf{Y}=\mathbf{Y}\cap\mathbf{Z}=\emptyset$ (but $\mathbf{X}$ and $\mathbf{Z}$ can overlap). $P(\mathbf{y}|\mathbf{z})$ is invariant under an intervention on $\mathbf{X}$ if*

(1) for every $X\in\mathbf{X}\cap\mathbf{Z}$, there is no d-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z}\backslash\{X\}$ that is into X;

(2) for every $X\in\mathbf{X}\cap(\mathbf{An}_{\mathcal{G}}(\mathbf{Z})\backslash\mathbf{Z})$, there is no d-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z}$; and

(3) for every $X\in\mathbf{X}\backslash\mathbf{An}_{\mathcal{G}}(\mathbf{Z})$, there is no d-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z}$ that is out of X.[17]

Remark: Because $\mathbf{Z}\subseteq\mathbf{An}_{\mathcal{G}}(\mathbf{Z})$, $\mathbf{X}\cap\mathbf{Z},\mathbf{X}\cap(\mathbf{An}_{\mathcal{G}}(\mathbf{Z})\backslash\mathbf{Z})$ and $\mathbf{X}\backslash\mathbf{An}_{\mathcal{G}}(\mathbf{Z})$ form a partition of $\mathbf{X}$. So for each member of $\mathbf{X}$, only one of the conditions is relevant.

The proposition is an equivalent formulation of Theorem 7.1 in Spirtes et al. (1993). It is not hard to check that the proposition follows from rules 2 and 3 in the DAG-based *do*-calculus (Proposition 7); the talk of d-separation in manipulated graphs is replaced by the talk of absence of d-connecting paths of certain orientations in the original graph. Conversely, the proposition implies the special case of rules 2 and 3 where the background intervention $do(\mathbf{U})$ is empty. Specifically, clause (1) in the proposition corresponds to rule 2 in the *do*-calculus; clauses (2) and (3) correspond to rule 3 in the *do*-calculus.

Spirtes et al. (1993, pp. 164-5) argued that these conditions are also "almost necessary" for invariance. What they meant is that if the conditions are not satisfied, then the causal structure does not *entail* the invariance, although there may exist some particular distribution compatible with the causal structure such that $P(\mathbf{y}|\mathbf{z})$ is invariant under some particular intervention on $\mathbf{X}$. From now on when we speak of invariance entailed by the causal DAG, we mean that the conditions in Proposition 18 are satisfied—or equivalently, that the invariance follows from an application of rule 2 or rule 3 in the DAG-based *do*-calculus.[18] Our purpose is to demonstrate that there are corresponding graphical

17. It is not hard to see that (3) is equivalent to saying that for every $X\in\mathbf{X}\backslash\mathbf{An}_{\mathcal{G}}(\mathbf{Z})$, there is no directed path from X to any member of $\mathbf{Y}$. Lemma 23 below is an immediate corollary of this equivalent formulation.

18. This stipulation is of course not intended to be a definition of the notion of *structurally entailed invariance*. A proper definition would be to the effect that for every distribution compatible with the causal structure, $P(\mathbf{y}|\mathbf{z})$ is invariant under any intervention of $\mathbf{X}$. The argument given by Spirtes et al. (1993, pp. 164-5) for (their equivalent formulation of) Proposition 18 suggests that the conditions are sufficient and necessary for structurally entailed invariance. Their argument uses the device of what they call policy variables, extra variables introduced into the

conditions relative to a PAG that are sufficient and necessary for the conditions in Proposition 18 to hold for each and every DAG compatible with the PAG.

Once again, we develop the conditions in two steps: first to MAGs and then to PAGs. In the first step, our goal is to find sufficient and necessary conditions for invariance entailed by a MAG, as defined below:

Definition 19 (Invariance entailed by a MAG) *Let $\mathcal{M}$ be a causal MAG over $\mathbf{O}$, and $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z} \subseteq \mathbf{O}$ be three sets of variables such that $\mathbf{X} \cap \mathbf{Y} = \mathbf{Y} \cap \mathbf{Z} = \emptyset$, $P(\mathbf{y}|\mathbf{z})$ is* **entailed to be invariant under interventions on X given** *$\mathcal{M}$ if for every DAG $\mathcal{G}(\mathbf{O}, \mathbf{L})$ represented by $\mathcal{M}$, $P(\mathbf{y}|\mathbf{z})$ is entailed to be invariant under interventions on $\mathbf{X}$ given $\mathcal{G}$ (i.e., the conditions in Proposition 18 are satisfied).*

The question is how to judge invariance entailed by a MAG without doing the intractable job of checking the conditions in Proposition 18 for each and every compatible DAG. The next few lemmas, Lemmas 20-23, state useful connections between d-connecting paths in a DAG and m-connecting paths in the corresponding MAG. Lemma 20 records the important result due to Richardson and Spirtes (2002) that d-separation relations among observed variables in a DAG with latent variables correspond exactly to m-separation relations in its MAG.

Lemma 20 *Let $\mathcal{G}$ be any DAG over $\mathbf{O} \cup \mathbf{L}$, and $\mathcal{M}$ be the MAG of $\mathcal{G}$ over $\mathbf{O}$. For any $A, B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, there is a path d-connecting A and B given $\mathbf{C}$ in $\mathcal{G}$ if and only if there is a path m-connecting A and B given $\mathbf{C}$ in $\mathcal{M}$.*

Proof This is a special case of Lemma 17 and Lemma 18 in Spirtes and Richardson (1996), and also a special case of Theorem 4.18 in Richardson and Spirtes (2002). ■

Given Lemma 20, we know how to tell whether clause (2) of Proposition 18 holds in all DAGs compatible with a given MAG. For the other two conditions in Proposition 18, we need to take into account the orientations of d-connecting paths.

Lemma 21 *Let $\mathcal{G}$ be any DAG over $\mathbf{O} \cup \mathbf{L}$, and $\mathcal{M}$ be the MAG of $\mathcal{G}$ over $\mathbf{O}$. For any $A, B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if there is a path d-connecting A and B given $\mathbf{C}$ in $\mathcal{G}$ that is into A, then there is a path m-connecting A and B given $\mathbf{C}$ in $\mathcal{M}$ that is either into A or contains an invisible edge out of A.*

Proof See Appendix B. ■

Lemma 22 *Let $\mathcal{M}$ be any MAG over $\mathbf{O}$. For any $A, B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if there is a path m-connecting A and B given $\mathbf{C}$ in $\mathcal{M}$ that is either into A or contains an invisible edge out of A, then there exists a DAG $\mathcal{G}$ over $\mathbf{O} \cup \mathbf{L}$ (for some extra variables $\mathbf{L}$) whose MAG is $\mathcal{M}$, such that in $\mathcal{G}$ there is a path d-connecting A and B given $\mathbf{C}$ that is into A.*

causal DAG to represent interventions. Given the causal DAG $\mathcal{G}$, a policy variable for a variable X is an (extra) parent of X but otherwise not adjacent to any other variables in $\mathcal{G}$. Interventions can then be simulated by conditioning on the intervention variables, and invariance can be reformulated as conditional independence involving intervention variables. The conditions in Proposition 18 are equivalent to saying that the variables in $\mathbf{Y}$ are d-separated from the policy variables for $\mathbf{X}$ by $\mathbf{Z}$ (in the graph augmented by the policy variables). It thus seems plausible that these conditions are sufficient and necessary for structurally entailed invariance, given that d-separation is a sufficient and necessary condition for structurally entailed conditional independence (Geiger et al., 1990; Meek, 1995b). But Spirtes et al. did not give a rigorous proof for necessity. As an anonymous reviewer points out, the rigorous proof, if any, would need to be carefully made, and in particular, one should be careful in treating policy variables as random variables. We will not take on this task here.

Proof See Appendix B. ■

Obviously these two lemmas are related to adapting clause (1) in Proposition 18 to MAGs. The next lemma is related to clause (3).

Lemma 23 *Let $\mathcal{G}$ be any DAG over $\mathbf{O} \cup \mathbf{L}$, and $\mathcal{M}$ be the MAG of $\mathcal{G}$ over $\mathbf{O}$. For any $A, B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain B or any descendant of A in $\mathcal{G}$ (or in $\mathcal{M}$, since $\mathcal{G}$ and $\mathcal{M}$ have the same ancestral relations among variables in $\mathbf{O}$), there is a path d-connecting A and B given $\mathbf{C}$ in $\mathcal{G}$ that is out of A if and only if there is a path m-connecting A and B given $\mathbf{C}$ in $\mathcal{M}$ that is out of A.*

Proof See Appendix B. ■

Given these lemmas, the conditions in Proposition 18 are readily translated into the following conditions for invariance given a MAG.

Theorem 24 *Suppose $\mathcal{M}$ is the causal MAG over a set of variables $\mathbf{O}$. For any $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{O}$, $\mathbf{X} \cap \mathbf{Y} = \mathbf{Y} \cap \mathbf{Z} = \emptyset$, $P(\mathbf{y}|\mathbf{z})$ is entailed to be invariant under interventions on $\mathbf{X}$ given $\mathcal{M}$ if and only if*

(1) for every $X \in \mathbf{X} \cap \mathbf{Z}$, there is no m-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z} \backslash \{X\}$ that is into X or contains an invisible edge out of X;

(2) for every $X \in \mathbf{X} \cap (\mathbf{An}_{\mathcal{M}}(\mathbf{Z}) \backslash \mathbf{Z})$, there is no m-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z}$; and

(3) for every $X \in \mathbf{X} \backslash \mathbf{An}_{\mathcal{M}}(\mathbf{Z})$, there is no m-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z}$ that is out of X.

Proof Given Lemma 21, if (1) holds, then for every DAG represented by $\mathcal{M}$, the first condition in Proposition 18 holds. Given Lemma 20 and the fact that $\mathcal{M}$ and all DAGs represented by $\mathcal{M}$ have the exact same ancestral relations among $\mathbf{O}$, if (2) holds, the second condition in Proposition 18 holds for every DAG represented by $\mathcal{M}$. Moreover, given Lemma 23, if (3) holds, the third condition in Proposition 18 holds for every DAG represented by $\mathcal{M}$. So (1), (2) and (3) together imply that $P(\mathbf{y}|\mathbf{z})$ is invariant under interventions on $\mathbf{X}$ given $\mathcal{M}$.

Conversely, if (1) fails, then by Lemma 22, there is a DAG represented by $\mathcal{M}$ in which the first condition in Proposition 18 fails. Likewise with conditions (2) and (3), in light of Lemmas 20 and 23 and the fact that $\mathcal{M}$ and a DAG represented by $\mathcal{M}$ have the exact same ancestral relations among $\mathbf{O}$. So (1), (2) and (3) are also necessary for $P(\mathbf{y}|\mathbf{z})$ to be entailed to be invariant under interventions on $\mathbf{X}$ given $\mathcal{M}$. ■

For example, given the MAG in Figure 3(a), $P(L|G,S)$ is invariant under interventions on S, because the only m-connecting path between L and S given G is $\langle L, S \rangle$, which contains a visible directed edge out of L, and so the relevant clause in Theorem 24, clause (1), is satisfied. By contrast, $P(L|G,S)$ is not entailed to be invariant under interventions on G given the MAG—in the sense that there exists a causal DAG compatible with the MAG given which $P(L|G,S)$ is not entailed to be invariant under interventions on G—because clause (1) is not satisfied.

In a similar fashion, we can extend the result to invariance entailed by a PAG. Definition first:

Definition 25 (Invariance entailed by a PAG) *Let $\mathcal{P}$ be a PAG over* $\mathbf{O}$*, and* $\mathbf{X},\mathbf{Y},\mathbf{Z} \subseteq \mathbf{O}$ *be three sets of variables such that* $\mathbf{X} \cap \mathbf{Y} = \mathbf{Y} \cap \mathbf{Z} = \emptyset$*, $P(\mathbf{y}|\mathbf{z})$ is* **entailed to be invariant under interventions on X given** *$\mathcal{P}$ if for every MAG $\mathcal{M}$ in the Markov equivalence class represented by $\mathcal{P}$, $P(\mathbf{y}|\mathbf{z})$ is entailed to be invariant under interventions on* $\mathbf{X}$ *given $\mathcal{M}$.*

We need a few lemmas that state connections between m-connecting paths in a MAG and definite m-connecting paths (as opposed to mere possibly m-connecting paths) in its PAG. By the definition of definite m-connecting paths (Definition 4), definite m-connection in a PAG implies m-connection in every MAG represented by the PAG. It is not obvious, however, that m-connection in a MAG will always be revealed as definite m-connection in its PAG. Fortunately, this turns out to be true. However, the proof is highly involved, and relies on many results about the properties of PAGs and the transformation between PAGs and MAGs presented in Zhang (2006, chapters 3-4), which would take up too much space and might distract readers from the main points of the present paper. So we will simply state the fact here, and refer interested readers to Zhang (2006, chap. 5, Lemma 5.1.7) for the proof.

Lemma 26 *Let $\mathcal{M}$ be a MAG over* $\mathbf{O}$*, and $\mathcal{P}$ be the PAG that represents $[\mathcal{M}]$. For any $A,B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if there is a path m-connecting A and B given* $\mathbf{C}$ *in $\mathcal{M}$, then there is a path definitely m-connecting A and B given* $\mathbf{C}$ *in $\mathcal{P}$. Furthermore, if there is an m-connecting path in $\mathcal{M}$ that is either into A or out of A with an invisible directed edge, then there is a definite m-connecting path in $\mathcal{P}$ that does not start with a definitely visible edge out of A.*

Proof See the proof of Lemma 5.1.7 in Zhang (2006, pp. 207). ■

The converse to the second part of Lemma 26 is also true.

Lemma 27 *Let $\mathcal{P}$ be a PAG over* $\mathbf{O}$*. For any $A,B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if there is a path definitely m-connecting A and B given* $\mathbf{C}$ *in $\mathcal{P}$ that does not start with a definitely visible edge out of A, then there exists a MAG $\mathcal{M}$ in the equivalence class represented by $\mathcal{P}$ in which there is a path m-connecting A and B given* $\mathbf{C}$ *that is either into A or includes an invisible directed edge out of A.*

Proof See Appendix B. ■

Lemmas 26 and 27 are useful for establishing conditions analogous to clauses (1) and (2) in Theorem 24. For clause (3), we need two more lemmas.

Lemma 28 *Let $\mathcal{M}$ be a MAG over* $\mathbf{O}$*, and $\mathcal{P}$ be the PAG that represents $[\mathcal{M}]$. For any $A,B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain B or any descendant of A in $\mathcal{M}$, if there is a path m-connecting A and B given* $\mathbf{C}$ *in $\mathcal{M}$ that is out of A, then there is a path definitely m-connecting A and B given* $\mathbf{C}$ *in $\mathcal{P}$ that is not into A (i.e., the edge incident to A on the path is either $A \circ\!\!-\!\!\circ$, or $A \circ\!\!\rightarrow$, or $A \rightarrow$).*

Proof See Appendix B. ■

Lemma 29 *Let $\mathcal{P}$ be a PAG over* $\mathbf{O}$*. For any $A,B \in \mathbf{O}$ and $\mathbf{C} \subseteq \mathbf{O}$ that does not contain A or B, if there is a path definitely m-connecting A and B given* $\mathbf{C}$ *in $\mathcal{P}$ that is not into A, then there exists a MAG $\mathcal{M}$ represented by $\mathcal{P}$ in which there is a path m-connecting A and B given* $\mathbf{C}$ *that is out of A.*

Proof See Appendix B. ■

The main theorem follows.

Theorem 30 *Suppose $\mathcal{P}$ is the causal PAG over a set of variables $\mathbf{O}$. For any $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{O}$ such that $\mathbf{X} \cap \mathbf{Y} = \mathbf{Y} \cap \mathbf{Z} = \emptyset$, $P(\mathbf{y}|\mathbf{z})$ is entailed to be invariant under interventions on $\mathbf{X}$ given $\mathcal{P}$ if and only if*

(1) for every $X \in \mathbf{X} \cap \mathbf{Z}$, every definite m-connecting path, if any, between X and any member of $\mathbf{Y}$ given $\mathbf{Z} \backslash \{X\}$ is out of X with a definitely visible edge;

(2) for every $X \in \mathbf{X} \cap (\mathbf{PossibleAn}_{\mathcal{P}}(\mathbf{Z}) \backslash \mathbf{Z})$, there is no definite m-connecting path between X and any member of $\mathbf{Y}$ given $\mathbf{Z}$; and

(3) for every $X \in \mathbf{X} \backslash \mathbf{PossibleAn}_{\mathcal{P}}(\mathbf{Z})$, every definite m-connecting path, if any, between X and any member of $\mathbf{Y}$ given $\mathbf{Z}$ is into X.

Proof We show that (1), (2) and (3) are sufficient and necessary for the corresponding conditions in Theorem 24 to hold for all MAGs represented by $\mathcal{P}$. It follows from Lemma 26 that if (1) holds, then the first condition in Theorem 24 holds for all MAGs represented by $\mathcal{P}$. Note moreover that for every MAG $\mathcal{M}$ represented by $\mathcal{P}$, $\mathbf{An}_{\mathcal{M}}(\mathbf{Z}) \subseteq \mathbf{PossibleAn}_{\mathcal{P}}(\mathbf{Z})$. It again follows from Lemma 26 that if (2) holds, then the second condition in Theorem 24 holds for all MAGs represented by $\mathcal{P}$. Finally, it follows from Lemma 28 (and Lemma 26) that if (3) holds, the third condition in Theorem 24 holds for all MAGs represented by $\mathcal{P}$. Hence (1), (2) and (3) are sufficient.

Conversely, if (1) fails, then by Lemma 27, there exists a MAG represented by $\mathcal{P}$ for which the first condition in Theorem 24 fails.

To show the necessity of (2), we need the fact mentioned in Footnote 11 that if X is a possible ancestor of a vertex $Z \in \mathbf{Z}$ in $\mathcal{P}$, then there exists a MAG represented by $\mathcal{P}$, in which X is an ancestor of Z. So if (2) fails, that is, there is a definite m-connecting path between a variable $X \in \mathbf{X} \cap (\mathbf{PossibleAn}_{\mathcal{P}}(\mathbf{Z}) \backslash \mathbf{Z})$ and a member of $\mathbf{Y}$ given $\mathbf{Z}$ in $\mathcal{P}$, then there exists a MAG $\mathcal{M}$ represented by $\mathcal{P}$ in which $X \in \mathbf{X} \cap (\mathbf{An}_{\mathcal{M}}(\mathbf{Z}) \backslash \mathbf{Z})$, and there is an m-connecting path between X and a member of $\mathbf{Y}$ given $\mathbf{Z}$, which violates clause (2) of Theorem 24.

Lastly, if (3) fails, that is, there is a definite m-connecting path between a variable $X \in \mathbf{X} \backslash \mathbf{PossibleAn}_{\mathcal{P}}(\mathbf{Z})$ and a member of $\mathbf{Y}$ given $\mathbf{Z}$ that is *not* into X, then it follows from Lemma 29 that there exists a MAG $\mathcal{M}$ represented by $\mathcal{P}$ in which there is an m-connecting path between X and a member of $\mathbf{Y}$ given $\mathbf{Z}$ that is out of X. Moreover, since $X \in \mathbf{X} \backslash \mathbf{PossibleAn}_{\mathcal{P}}(\mathbf{Z})$, X cannot be an ancestor of $\mathbf{Z}$ in $\mathcal{M}$, that is, $X \in \mathbf{X} \backslash \mathbf{An}_{\mathcal{M}}(\mathbf{Z})$. So $\mathcal{M}$ fails clause (3) of Theorem 24. Therefore, the conditions are also necessary. ■

For a simple illustration, consider again the PAG in Figure 4. Given the PAG, it can be inferred that $P(L|G,S)$ is invariant under interventions on I, because there is no definite m-connecting path between L and I given $\{G,S\}$, satisfying the relevant clause—clause (2)—in Theorem 30. $P(L|G,S)$ is also invariant under interventions on S because the only definitely m-connecting path between L and S given $\{G\}$ is $S \rightarrow L$ which contains a definitely visible edge out of S, satisfying the relevant clause—clause (1)—in Theorem 30.

On the other hand, for example, $P(S)$ is not entailed to be invariant under interventions on I. Note that given the MAG of Figure 3(b), $P(S)$ is indeed entailed to be invariant under interventions on I, but this invariance is not unanimously implied in the equivalence class. Given some other MAGs in the class, such as the one in Figure 3(a), $P(S)$ is not entailed to be invariant under interventions on I.

As briefly noted in the last section, the PAG-based *do*-calculus in its current form is not complete. We mentioned three issues that might be responsible for this (cf. the comments right after Theorem 17), but only one of them we are sure leads to counterexamples—examples in which a rule in the DAG-based calculus is applicable for all DAGs compatible with the given PAG, but the corresponding rule in the PAG-based calculus is not applicable. It is the fact that the calculus is formulated in terms of absence of possibly m-connecting paths. Consider the example we used to illustrate the difference between definite and possibly m-connecting paths in Section 3. Given the PAG in Figure 5, we cannot apply rule 2 of the PAG-based *do*-calculus to conclude that $P(W|do(X),Y,Z) = P(W|Y,Z)$, because there is a possibly m-connecting path between X and W relative to $\{Y,Z\}$ in the PAG (note that since $X \in \mathbf{PossibleAn}(\{Y,Z\})$, the rule does not require manipulating the graph). However, it can be shown that for every DAG compatible with the PAG, X and W are d-separated by $\{Y,Z\}$ in either the X-upper-manipulation of the DAG or in the DAG itself. So rule 2 of the DAG-based *do*-calculus is actually applicable given any DAG compatible with the PAG.

Although we suspect that such counterexamples may not be encountered often in practice, it is at least theoretically interesting to handle them. Our results in this section provide an improvement in regard to the important special case of invariance. That is, the conditions given in Theorem 30 are complete for deriving statements of invariance, in the following sense: if the conditions therein fail relative to a PAG, then there exists a DAG represented by the PAG given which the conditions in Proposition 18 do not hold. The example in Figure 5 is not a counterexample to the completeness of Theorem 30. Unlike the *do*-calculus presented in Theorem 17, Theorem 30 implies that $P(W|Y,Z)$ is entailed to be invariant under interventions on X given the PAG (and hence we can conclude that $P(W|do(X),Y,Z) = P(W|Y,Z)$), because there is no definite m-connecting path between X and W relative to $\{Y,Z\}$ in the PAG. Whether it is valid to formulate the PAG-based *do*-calculus in terms of definite m-connecting paths is an open question at this point (cf. Footnote 15).[19]

Theorem 30 is in style very similar to Theorems 7.3 and 7.4 in Spirtes et al. (1993). The latter are formulated with respect to a *partially oriented inducing path graph* (POIPG). We include in Appendix A a description of the inducing path graphs (IPGs) as well as their relationship to ancestral graphs. As shown there, syntactically the class of ancestral graphs is a proper subclass of the class of inducing path graphs. In consequence a PAG in general reveals more qualitative causal information than a POIPG. In addition, it seems MAGs are easier to parameterize than IPGs. (For a linear parametrization of MAGs, see Richardson and Spirtes, 2002.)

Apart from the advantages of working with MAGs and PAGs over IPGs and POIPGs, our Theorem 30 is superior to Spirtes et al.'s theorems in that our theorem is formulated in terms of definite m-connecting paths, whereas theirs, like the results in the last section, are formulated in terms of

19. Here is another way to view the open problem. As explained earlier, *do*-calculus is essentially a generalization of the invariance conditions. Not only does it address the question of when $(\mathbf{y}|\mathbf{z})$ is invariant under an intervention $\mathbf{X} := \mathbf{x}$, it also addresses the more general question of when a post-intervention probability $P_{\mathbf{u}}(\mathbf{y}|\mathbf{z})$ would be invariant under a *further* intervention $\mathbf{X} := \mathbf{x}$. Our results in this section do not cover the latter question. To generalize the results in terms of definite m-connecting paths to address the latter question is parallel to improving the *do*-calculus.

possibly m-connecting paths. As a result, their conditions are only sufficient but not necessary. Regarding the case in Figure 5, for example, their theorems do not imply that $P(W|Y,Z)$ is entailed to be invariant under interventions on X, due to the presence of the possibly m-connecting path in the graph (which in this case is also the POIPG). Furthermore, since definite m-connecting paths are special cases of possibly m-connecting paths, there are more possibly m-connecting paths than definite m-connecting paths to check in a PAG. This may turn out to be a computational advantage for our theorem.

6. Conclusion

Causal reasoning about consequences of interventions has received rigorous and interesting treatments in the framework of causal Bayesian networks. Much of the work assumes that the structure of the causal Bayesian network, represented by a directed acyclic graph, is fully given. In this paper we have provided some results about causal reasoning under weaker causal assumptions, represented by a maximal ancestral graph or a partial ancestral graph, the latter of which is fully testable with observational data (assuming the causal Faithfulness condition).

Theorem 17 in Section 4 gives us a *do*-calculus under testable causal assumptions, represented by a PAG. The idea is that when any rule in the calculus is applicable given the PAG, the corresponding rule in Pearl's original *do*-calculus is applicable relative to each and every DAG compatible with the PAG. The converse, however, is not true; it is not the case that whenever all DAGs compatible with the PAG sanction the application of a certain rule in the *do*-calculus, the corresponding rule in the PAG-based calculus is also applicable. An interesting project is to either improve the calculus, or to investigate more closely the extent to which the current version is not complete.

As a first step towards improvement, we examined in Section 5 an important special case of the *do*-calculus—the graphical conditions for invariance under interventions—and presented sufficient and necessary conditions for invariance given a PAG. These conditions are very similar but also superior to the analogous results proved by Spirtes et al. (1993). In the latter work, there is also an algorithm (named Prediction Algorithm) for identifying post-intervention probabilities based on the conditions for invariance. The results in this paper can certainly be used to improve that algorithm.

The search for a syntactic derivation in the *do*-calculus to identify a post-intervention probability is no minor computational task. For this reason, it is worth deriving handy graphical criteria for identifiability from the *do*-calculus. Since invariant quantities are the most basic identifiable quantities, the condition for invariance is the most basic among such graphical criteria. Other graphical criteria in the literature, including the well known "back door criterion" and "front door criterion", should be extendible to PAGs in the same way as we did for invariance. On the other hand, a novel approach to identification has been developed recently by Tian and Pearl (2004), which proves computationally attractive. To adapt that approach to ancestral graphs is probably a worthy project.

Acknowledgments

I am grateful to Clark Glymour, Thomas Richardson, and Peter Spirtes for their helpful comments on the part of my dissertation this paper is based on. Thanks also to three anonymous referees for helping improve the paper significantly. One of them, especially, made extremely detailed and helpful suggestions.

Appendix A. Inducing Path Graphs

The theory of invariance under interventions developed in this paper is largely parallel to that developed in Spirtes et al. (1993). Their theory is based on a graphical representation called inducing path graphs. This graphical object is not given an independent syntactic definition, but defined via a construction relative to a DAG (with latent variables). It is clear from the construction that this representation is closely related to MAGs. In this appendix we specify the exact relationship between them. In particular, we justify an independent syntactic definition of inducing path graphs, which makes it clear that syntactically the class of MAGs is a subclass of inducing path graphs.

An *inducing path graph (IPG)* is a directed mixed graph, defined relative to a DAG, through the following construction:

Input: a DAG $\mathcal{G}$ over $\langle \mathbf{O}, \mathbf{L} \rangle$
Output: an IPG $I_{\mathcal{G}}$ over $\mathbf{O}$

1. for each pair of variables $A, B \in \mathbf{O}$, A and B are adjacent in $I_{\mathcal{G}}$ if and only if there is an inducing path between them relative to $\mathbf{L}$ in $\mathcal{G}$;

2. for each pair of adjacent vertices A, B in $I_{\mathcal{G}}$, mark the A-end of the edge as an arrowhead if there is an inducing path between A and B that is into A, otherwise mark the A-end of the edge as a tail.

It can be shown that the construction outputs a mixed graph $I_{\mathcal{G}}$ in which the set of m-separation relations matches exactly the set of d-separation relations among $\mathbf{O}$ in the original DAG $\mathcal{G}$ (Spirtes and Verma, 1992). Furthermore, $I_{\mathcal{G}}$ encodes information about inducing paths in the original graph, which in turn implies features of the original DAG that bear causal significance. Specifically, we have two useful facts: (i) if there is an inducing path between A and B relative to $\mathbf{L}$ that is out of A, then A is an ancestor of B in $\mathcal{G}$; (ii) if there is an inducing path between A and B relative to $\mathbf{L}$ that is into both A and B, then A and B have a common ancestor in $\mathbf{L}$ unmediated by any other observed variable.[20] So $I_{\mathcal{G}}$, just like the MAG for $\mathcal{G}$, represents both the conditional independence relations and (features of) the causal structure among the observed variables $\mathbf{O}$. Since the above construction produces a unique graph given a DAG $\mathcal{G}$, it is fair to call $I_{\mathcal{G}}$ the IPG for $\mathcal{G}$.

Therefore a directed mixed graph over a set of variables is an IPG if it is the IPG for some DAG. We now show that a directed mixed graph is an IPG if and only if it is maximal and does not contain a directed cycle.

Theorem 31 *For any directed mixed graph I over a set of variables $\mathbf{O}$, there exists a DAG $\mathcal{G}$ over $\mathbf{O}$ and possibly some extra variables $\mathbf{L}$ such that $I = I_{\mathcal{G}}$—that is, I is the IPG for $\mathcal{G}$—if and only if*

(i1) There is no directed cycle in I; and

(i2) I is maximal (i.e., there is no inducing path between two non-adjacent variables).

Proof We first show that the conditions are necessary (**only if**). Suppose there exists a DAG $\mathcal{G}(\mathbf{O}, \mathbf{L})$ whose IPG is I. In other words, I is the output of the IPG construction procedure given $\mathcal{G}$. If there is any directed cycle in I, say $c = \langle O_1, \ldots, O_n, O_1 \rangle$, then between any pair of adjacent nodes in the cycle, O_i and O_{i+1} ($1 \leq i \leq n$ and $O_{n+1} = O_1$), there is an inducing path between them in $\mathcal{G}$ relative

20. For more details of the causal interpretation of IPGs, see Spirtes et al. (1993, pp. 130-138).

to **L**, which, by one of the facts mentioned earlier, implies that O_i is an ancestor of O_{i+1} in $\mathcal{G}$. Thus there would be a directed cycle in $\mathcal{G}$ as well, a contradiction. Therefore there is no directed cycle in I. To show that it is also maximal, consider any two non-adjacent nodes A and B in I. We show that there is no inducing path in I between A and B. Otherwise let $p = \langle A, O_1, \ldots, O_n, B \rangle$ be an inducing path. By the construction, there is an inducing path relative to **L** in $\mathcal{G}$ between A and O_1 that is into O_1, and an inducing path relative to **L** in $\mathcal{G}$ between B and O_n that is into O_n, and for every $1 \leq i \leq i-1$, there is an inducing path relative to **L** in $\mathcal{G}$ between O_i and O_{i+1} that is into both. By Lemma 32 in Appendix B, it follows that there is an inducing path between A and B relative to **L** in $\mathcal{G}$, and so A and B should be adjacent in I, a contradiction. Therefore I is also maximal.

Next we demonstrate sufficiency (**if**). If the two conditions hold, construct a DAG $\mathcal{G}$ as follows: retain all the directed edges in I, and for each bi-directed edge $A \leftrightarrow B$ in I, introduce a latent variable L_{AB} in $\mathcal{G}$ and replace $A \leftrightarrow B$ with $A \leftarrow L_{AB} \rightarrow B$.[21] It is easy to see that the resulting graph $\mathcal{G}$ is a DAG, as in I there is no directed cycle. We show that $I = I_{\mathcal{G}}$, the IPG for $\mathcal{G}$. For any pair of variables A and B in I, there are four cases to consider:

Case 1: $A \rightarrow B$ is in I. Then $A \rightarrow B$ is also in $\mathcal{G}$, so A and B are adjacent in $I_{\mathcal{G}}$. In $I_{\mathcal{G}}$, the edge between A and B is not $A \leftarrow B$, because otherwise B would have to be an ancestor of A in $\mathcal{G}$, a contradiction. The edge is not $A \leftrightarrow B$ either, because otherwise there would have to be a latent variable that is a parent of both A and B, which by the construction of $\mathcal{G}$ is not the case. So $A \rightarrow B$ is also in $I_{\mathcal{G}}$.

Case 2: $A \leftarrow B$ is in I. By the same argument as in *Case 1*, $A \leftarrow B$ is also in $I_{\mathcal{G}}$.

Case 3: $A \leftrightarrow B$ is in I. Then there is a L_{AB} such that $A \leftarrow L_{AB} \rightarrow B$ is in $\mathcal{G}$. Then obviously $\langle A, L_{AB}, B \rangle$ is an inducing path relative to **L** in $\mathcal{G}$ that is into both A and B, and hence $A \leftrightarrow B$ is also in $I_{\mathcal{G}}$.

Case 4: A and B are not adjacent in I. We show that they are not adjacent in $I_{\mathcal{G}}$ either. For this, we only need to show that there is no inducing path between A and B relative to **L** in $\mathcal{G}$. Suppose otherwise that there is such an inducing path p between A and B in $\mathcal{G}$. Let $\langle A, O_1, \ldots, O_n, B \rangle$ be the sub-sequence of p consisting of all observed variables on p. By the definition of inducing path, all O_i's ($1 \leq i \leq n$) are colliders on p and are ancestors of either A or B. By the construction of $\mathcal{G}$, it is easy to see that O_i's are also ancestors of either A or B in I. It is also easy to see that either $A \rightarrow O_1$ or $A \leftarrow L_{AO_1} \rightarrow O_1$ appears in $\mathcal{G}$, which implies that there is an edge between A and O_1 that is into O_1 in I. Likewise, there is an edge between O_n and B that is into O_n in I, and there is an edge between O_i and O_{i+1} that is into both in I for all $1 \leq i \leq n-1$. So $\langle A, O_1, \ldots, O_n, B \rangle$ constitutes an inducing path between A and B in I, which contradicts the assumption that I is maximal. So there is no inducing path between A and B relative to **L** in $\mathcal{G}$, which means that A and B are not adjacent in $I_{\mathcal{G}}$.

Therefore $I = I_{\mathcal{G}}$, the IPG for $\mathcal{G}$. ■

Given this theorem, it is clear that we can define IPGs in terms of $(i1)$ and $(i2)$. So a MAG is also an IPG, but an IPG is not necessarily a MAG, as the former may contain an almost directed cycle. The simplest IPG which is not a MAG is shown in Figure 11.

Spirtes et al. (1993) uses *partially oriented inducing path graphs (POIPGs)* to represent Markov equivalence classes of IPGs. The idea is exactly the same as PAGs. A (complete) POIPG displays (all) common marks in a Markov equivalence class of IPGs. An obvious fact is that given a set of conditional independence facts that admits a faithful representation by a MAG, the Markov equiva-

21. This is named *canonical DAG* in Richardson and Spirtes (2002).

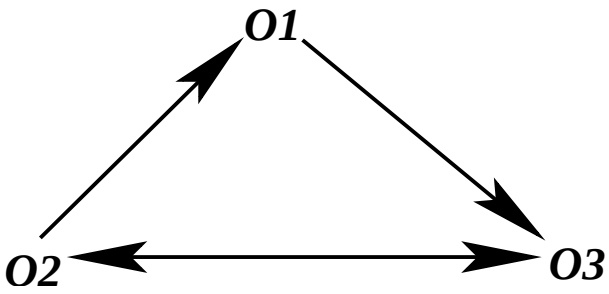


Figure 11: A simplest IPG that is not a MAG

lence class of MAGs is included in the Markov equivalence class of IPGs. It follows that the POIPG cannot contain more informative marks than the PAG, but may contain fewer. So a PAG usually reveals more qualitative causal information than a POIPG does.

Appendix B. Proofs of the Lemmas

In proving some of the lemmas, we will use the following fact, which was proved in, for example, Spirtes et al. (1999, pp. 243):

Lemma 32 *Let $\mathcal{G}(\mathbf{O},\mathbf{L})$ be a DAG, and $\langle V_0,\ldots,V_n\rangle$ be a sequence of distinct variables in $\mathbf{O}$. If (1) for all $0 \le i \le n-1$, there is an inducing path in $\mathcal{G}$ between V_i and V_{i+1} relative to $\mathbf{L}$ that is into V_i unless possibly $i = 0$ and is into V_{i+1} unless possibly $i = n-1$; and (2) for all $1 \le i \le n-1$, V_i is an ancestor of either V_0 or V_n in $\mathcal{G}$; then there is a subpath s of the concatenation of those inducing paths that is an inducing path between V_0 and V_n relative to $\mathbf{L}$ in $\mathcal{G}$. Furthermore, if the said inducing path between V_0 and V_1 is into V_0, then s is into V_0, and if the said inducing path between V_{n-1} and V_n is into V_n, then s is into V_n.*

Proof This is a special case of Lemma 10 in Spirtes et al. (1999, pp. 243). See their paper for a detailed proof. (One may think that the concatenation itself would be an inducing path between V_0 and V_n. This is almost correct, except that the concatenation may contain the same vertex multiple times. So in general it is a subsequence of the concatenation that constitutes an inducing path between V_0 and V_n.) ■

Lemma 32 gives a way to argue for the presence of an inducing path between two variables in a DAG, and hence is very useful for demonstrating that two variables are adjacent in the corresponding MAG. We will see several applications of this lemma in the subsequent proofs.

Proof of Lemma 9

Proof Since there is an inducing path between A and B relative to $\mathbf{L}$ in $\mathcal{G}$, A and B are adjacent in $\mathcal{M}$. Furthermore, since $A \in \mathbf{An}_{\mathcal{G}}(B)$, the edge between A and B in $\mathcal{M}$ is $A \to B$. We now show that it is invisible in $\mathcal{M}$. To show this, it suffices to show that for any C, if in $\mathcal{M}$ there is an edge between C and A that is into A or there is a collider path between C and A that is into A and every vertex on the path is a parent of B, then C is adjacent to B, which means that the condition for visibility cannot be met.

Let u be an inducing path between A and B relative to $\mathbf{L}$ in $\mathcal{G}$ that is into A. For any C, we consider the two possible cases separately:

Case 1: There is an edge between C and A in $\mathcal{M}$ that is into A. Then, by the way $\mathcal{M}$ is constructed from $\mathcal{G}$, there must be an inducing path u' in $\mathcal{G}$ between A and C relative to $\mathbf{L}$. Moreover, u' is into A, for otherwise A would be an ancestor of C, so that the edge between A and C in $\mathcal{M}$ would be out of A. Given u, u' and the fact that $A \in \mathbf{An}_{\mathcal{G}}(B)$, we can apply Lemma 32 to conclude that there is an inducing path between C and B relative to $\mathbf{L}$ in $\mathcal{G}$, which means C and B are adjacent in $\mathcal{M}$.

Case 2: There is a collider path p in $\mathcal{M}$ between C and A that is into A and every non-endpoint vertex on the path is a parent of B. For every pair of adjacent vertices $\langle V_i, V_{i+1}\rangle$ on p, the edge is $V_i \leftrightarrow V_{i+1}$ if $V_i \neq C$, and otherwise either $C \leftrightarrow V_{i+1}$ or $C \rightarrow V_{i+1}$. It follows that there is an inducing path in $\mathcal{G}$ between V_i and V_{i+1} relative to $\mathbf{L}$ such that the path is into V_{i+1}, and is into V_i unless possibly $V_i = C$. Given these inducing paths and the fact that every variable other than C on p is an ancestor of B, we can apply Lemma 32 to conclude that there is an inducing path between C and B relative to $\mathbf{L}$ in $\mathcal{G}$, which means C and B are adjacent in $\mathcal{M}$.

Therefore, the edge $A \rightarrow B$ is invisible in $\mathcal{M}$. ■

Proof of Lemma 10

Proof Construct a DAG from $\mathcal{M}$ as follows:

1. Leave every directed edge in $\mathcal{M}$ as it is. Introduce a latent variable L_{AB} and add $A \leftarrow L_{AB} \rightarrow B$ to the graph.

2. for every bi-directed edge $Z \leftrightarrow W$ in $\mathcal{M}$, delete the bi-directed edge. Introduce a latent variable L_{ZW} and add $Z \leftarrow L_{ZW} \rightarrow W$ to the graph.

The resulting graph we denote by $\mathcal{G}$, a DAG over $(\mathbf{O}, \mathbf{L})$, where $\mathbf{L} = \{L_{AB}\} \cup \{L_{ZW} | Z \leftrightarrow W \text{ is in } \mathcal{M}\}$. Obviously $\mathcal{G}$ is a DAG in which A and B share a latent parent. We need to show that $\mathcal{M} = \mathcal{M}_{\mathcal{G}}$, that is, $\mathcal{M}$ is the MAG of $\mathcal{G}$. For any pair of variables X and Y, there are four cases to consider:

Case 1: $X \rightarrow Y$ is in $\mathcal{M}$. Since $\mathcal{G}$ retains all directed edges in $\mathcal{M}$, $X \rightarrow Y$ is also in $\mathcal{G}$, and hence is also in $\mathcal{M}_{\mathcal{G}}$.

Case 2: $X \leftarrow Y$ is in $\mathcal{M}$. Same as *Case 1*.

Case 3: $X \leftrightarrow Y$ is in $\mathcal{M}$. Then there is a latent variable L_{XY} in $\mathcal{G}$ such that $X \leftarrow L_{XY} \rightarrow Y$ appears in $\mathcal{G}$. Since $X \leftarrow L_{XY} \rightarrow Y$ is an inducing path between X and Y relative to $\mathbf{L}$ in $\mathcal{G}$, X and Y are adjacent in $\mathcal{M}_{\mathcal{G}}$. Furthermore, it is easy to see that the construction of $\mathcal{G}$ does not create any directed path from X to Y or from Y to X. So X is not an ancestor of Y and Y is not an ancestor of X in $\mathcal{G}$. It follows that in $\mathcal{M}_{\mathcal{G}}$ the edge between X and Y is $X \leftrightarrow Y$.

Case 4: X and Y are not adjacent in $\mathcal{M}$. We show that in $\mathcal{G}$ there is no inducing path between X and Y relative to $\mathbf{L}$. Suppose otherwise that there is one. Let p be an inducing path between X and Y relative to $\mathbf{L}$ in $\mathcal{G}$ that includes a minimal number of observed variables. Let $\langle X, O_1, \ldots, O_n, Y\rangle$ be the sub-sequence of p consisting of all observed variables on p. By the definition of inducing path, all O_i's ($1 \leq i \leq n$) are colliders on p and are ancestors of either X or Y in $\mathcal{G}$. Since the construction of $\mathcal{G}$ does not create any new directed path from an observed variable to another observed variable, O_i's are also ancestors of either X or Y in $\mathcal{M}$. Since O_1 is a collider on p, either $X \rightarrow O_1$ or $X \leftarrow L_{XO_1} \rightarrow O_1$ appears in $\mathcal{G}$. Either way there is an edge between X and O_1 that is into O_1 in $\mathcal{M}$. Likewise, there is an edge between O_n and Y that is into O_n in $\mathcal{M}$.

Moreover, for all $1 \leq i \leq n-1$, the path p in $\mathcal{G}$ contains $O_i \leftarrow L_{O_iO_{i+1}} \rightarrow O_{i+1}$, because all O_i's are colliders on p. Thus in $\mathcal{M}$ there is an edge between O_i and O_{i+1}. Regarding these edges, by construction of the MAG, either all of them are bi-directed, or one of them is $A \rightarrow B$ and others are bi-directed. In the former case, $\langle X, O_1, \ldots, O_n, Y \rangle$ constitutes an inducing path between X and Y in $\mathcal{M}$, which contradicts the maximality of $\mathcal{M}$. In the latter case, without loss of generality, suppose $\langle A, B \rangle = \langle O_k, O_{k+1} \rangle$. Then $\langle X, O_1, ..., O_k = A \rangle$ is a collider path into A in $\mathcal{M}$. We now show by induction that for all $1 \leq i \leq k-1$, O_i is a parent of B in $\mathcal{M}$.

Consider O_{k-1} in the base case. O_{k-1} is adjacent to B, for otherwise $A \rightarrow B$ would be visible in $\mathcal{M}$ because there is an edge between O_{k-1} and A that is into A. The edge between O_{k-1} and B is not $O_{k-1} \leftarrow B$, for otherwise there would be $A \rightarrow B \rightarrow O_{k-1}$ and yet an edge between O_{k-1} and A that is into A in $\mathcal{M}$, which contradicts the fact that $\mathcal{M}$ is ancestral. The edge between them is not $O_{k-1} \leftrightarrow B$, for otherwise there would be an inducing path between X and Y relative to $\mathbf{L}$ in $\mathcal{G}$ that includes fewer observed variables than p does, which contradicts our choice of p. So O_{k-1} is a parent of B in $\mathcal{M}$.

In the inductive step, suppose for all $1 < m+1 \leq j \leq k-1$, O_j is a parent of B in $\mathcal{M}$, and we need to show that O_m is also a parent of B in $\mathcal{M}$. The argument is essentially the same as in the base case. Specifically, O_m and B are adjacent because otherwise it follows from the inductive hypothesis that $A \rightarrow B$ is visible. The edge is not $O_m \leftarrow B$ on pain of making $\mathcal{M}$ non-ancestral; and the edge is not $O_m \leftrightarrow B$ on pain of creating an inducing path that includes fewer observed variables than p does. So O_m is also a parent of B.

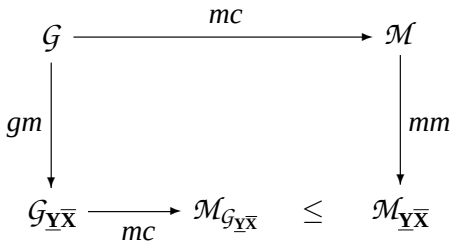
Now we have shown that for all $1 \leq i \leq k-1$, O_i is a parent of B in $\mathcal{M}$. It follows that X is adjacent to B, for otherwise $A \rightarrow B$ would be visible. Again, the edge is not $X \leftarrow B$ on pain of making $\mathcal{M}$ non-ancestral. So the edge between X and B in $\mathcal{M}$ is into B, but then there is an inducing path between X and Y relative to $\mathbf{L}$ in $\mathcal{G}$ that includes fewer observed variables than p does, which is a contradiction with our choice of p.

So our initial supposition is false. There is no inducing path between X and Y relative to $\mathbf{L}$ in $\mathcal{G}$, and hence X and Y are not adjacent in $\mathcal{M}_{\mathcal{G}}$.

Therefore $\mathcal{M} = \mathcal{M}_{\mathcal{G}}$. ■

Proof of Lemma 12

Proof Recall the diagram in Figure 7:



What we need to show is that $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is an I-map of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, or in other words, whatever m-separation relation is true in the former is also true in the latter. To show this, it suffices to show that $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is Markov equivalent to a supergraph of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$.

For that purpose, we first establish two facts: (1) every directed edge in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$ is also in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$; and (2) for every bi-directed edge $S \leftrightarrow T$ in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, S and T are also adjacent in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$; and the edge between S and T is either a bi-directed edge or an invisible directed edge in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$.

(1) is relatively easy to show. Note that for any $P \to Q$ in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, $P \notin \mathbf{Y}$, for otherwise P would not be an ancestor of Q in $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, and hence would not be a parent of Q in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$; and likewise $Q \notin \mathbf{X}$, for otherwise Q would not be a descendant of P in $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, and hence would not be a child of P in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$. Furthermore, because $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is a subgraph of $\mathcal{G}$, any inducing path between P and Q relative to $\mathbf{L}$ in $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is also present in $\mathcal{G}$, and any directed path from P to Q in the former is also present in the latter. This entails that $P \to Q$ is also in $\mathcal{M}$, the MAG of $\mathcal{G}$. Since $P \notin \mathbf{Y}$ and $Q \notin \mathbf{X}$, $P \to Q$ is also present in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. So (1) is true.

(2) is less obvious. First of all, note that if $S \leftrightarrow T$ is in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, then there is an inducing path between S and T relative to $\mathbf{L}$ in $\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ that is into both S and T. This implies that $S, T \notin \mathbf{X}$, and moreover there is also an inducing path between S and T relative to $\mathbf{L}$ in $\mathcal{G}$ that is into both S and T. Hence there is an edge between S and T in $\mathcal{M}$, the MAG of $\mathcal{G}$. The edge in $\mathcal{M}$ is either $S \leftrightarrow T$ or, by Lemma 9, an invisible directed edge ($S \leftarrow T$ or $S \to T$).

Because $S, T \notin \mathbf{X}$, if $S \leftrightarrow T$ appears in $\mathcal{M}$, it also appears in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. If, on the other hand, the edge between S and T in $\mathcal{M}$ is directed, suppose without loss of generality that it is $S \to T$. Either $S \in \mathbf{Y}$, in which case we have $S \leftrightarrow T$ in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, because $S \to T$ is invisible in $\mathcal{M}$; or $S \notin \mathbf{Y}$, and $S \to T$ remains in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. In the latter case we need to show that $S \to T$ is still invisible in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. Suppose for the sake of contradiction that $S \to T$ is visible in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, that there is a vertex R not adjacent to T, such that either $R \ast\!\!\to S$ is in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ or there is a collider path c in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ between R and S that is into S on which every collider is a parent of T. We show that $S \to T$ is also visible in $\mathcal{M}$. Consider the two possible cases separately:

Case 1: $R \ast\!\!\to S$ is in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. If the edge is $R \to S$, it is also in $\mathcal{M}$, because manipulations of a MAG do not create new directed edges. We now show that R and T are not adjacent in $\mathcal{M}$. Suppose otherwise. The edge between R and T has to be $R \to T$ in $\mathcal{M}$. Note that $R \notin \mathbf{Y}$ for otherwise $R \to S$ would be deleted or changed into a bi-directed edge; and we have already shown that $T \notin \mathbf{X}$. It follows that $R \to T$ would be present in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ as well, a contradiction. Hence R and T are not adjacent in $\mathcal{M}$, and so the edge $S \to T$ is also visible in $\mathcal{M}$.

Suppose, on the other hand, the edge between R and S in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is $R \leftrightarrow S$. In $\mathcal{M}$ the edge is either (i) $R \leftrightarrow S$, or (ii) $R \to S$. (It can't be $R \leftarrow S$ because then $S \in \mathbf{Y}$ and the edge $S \to T$ would not remain in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$.)

If (i) is the case, we argue that R and T are not adjacent in $\mathcal{M}$. Since $R \leftrightarrow S \to T$ is in $\mathcal{M}$, if R and T are adjacent, it has to be $R \leftrightarrow T$ or $R \to T$. In the former case, $R \leftrightarrow T$ would still be present in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ (because obviously $R, T \notin \mathbf{X}$), which is a contradiction. In the latter case, $R \to T$ is invisible in $\mathcal{M}$, for otherwise it is easy to see that $S \to T$ would also be visible. So either $R \to T$ remains in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ (if $R \notin \mathbf{Y}$), or it turns into $R \leftrightarrow T$ (if $R \in \mathbf{Y}$). In either case R and T would still be adjacent in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, a contradiction. Hence R and T are not adjacent in $\mathcal{M}$, and so the edge $S \to T$ is also visible in $\mathcal{M}$.

If (ii) is the case, then either R and T are not adjacent in $\mathcal{M}$, in which case $S \to T$ is also visible in $\mathcal{M}$; or R and T are adjacent in $\mathcal{M}$, in which case we now show that $S \to T$ is still visible. The edge between R and T in $\mathcal{M}$ has to be $R \to T$ (in view of $R \to S \to T$). Since R and T are not adjacent in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, and $R \to S$ is turned into $R \leftrightarrow S$ in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, by the definition of lower-manipulation (Definition 11), $R \to T$ is visible but $R \to S$ is invisible in $\mathcal{M}$. Because $R \to T$ is visible, by definition, there is a vertex Q not adjacent to T such that $Q \ast\!\!\to R$ is in $\mathcal{M}$ or there is a collider path in $\mathcal{M}$ between Q

and R that is into R on which every collider is a parent of T. But $R \to S$ is not visible, from which we can derive that $S \to T$ is visible in $\mathcal{M}$. Here is a sketch of the argument. If $Q \ast\!\!\to R$ is in $\mathcal{M}$, then Q and S must be adjacent (since otherwise $R \to S$ would be visible). It is then easy to derive that the edge between Q and S must be into S, which makes $S \to T$ visible. On the other hand, suppose there is a collider path c into R on which every collider is a parent of T. Then if there is a collider P on c such that $P \leftrightarrow S$ is in $\mathcal{M}$, we immediately get a collider path between Q and S that is into S on which every collider is a parent of T. This path makes $S \to T$ visible. Finally, if no collider on the path is a spouse of S, it is not hard to show that in order for $R \to S$ to be invisible, there has to be an edge between Q and S that is into S, which again makes $S \to T$ visible.

Case 2: There is a collider path c in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ between R and S that is into S on which every collider is a parent of T. We claim that every arrowhead on c, except possibly one at R, is also in $\mathcal{M}$. Because if an arrowhead is added at a vertex Q (which could be S) on c by the lower-manipulation, then $Q \in \mathbf{Y}$, but then the edge $Q \to T$ would not remain in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, a contradiction. So c is also a collider path in $\mathcal{M}$ that is into S. Furthermore, no new directed edges are introduced by lower-manipulation or upper-manipulation, so every collider on c is still a parent of T in $\mathcal{M}$.

It follows that if R and T are not adjacent in $\mathcal{M}$, then $S \to T$ is visible in $\mathcal{M}$. On the other hand, if R and T are adjacent in $\mathcal{M}$, it is either $R \leftrightarrow T$ or $R \to T$. Note that this edge is deleted in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. This implies that it is not $R \leftrightarrow T$ in $\mathcal{M}$: otherwise, the edge incident to R on c has to be bi-directed as well (since otherwise $\mathcal{M}$ is not ancestral), and hence if $R \leftrightarrow T$ is deleted, either the edge incident to R on c or the edge $S \to T$ should be deleted in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, which is a contradiction. So the edge is $R \to T$ in $\mathcal{M}$. Since $T \notin \mathbf{X}$ (for otherwise $S \to T$ would be deleted), $R \in \mathbf{Y}$, and $R \to T$ is visible in $\mathcal{M}$. But then it is easy to see that $S \to T$ is also visible in $\mathcal{M}$.

To summarize, we have shown that if $S \to T$ is visible in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, it is also visible in $\mathcal{M}$. Since it is not visible in $\mathcal{M}$, it is invisible in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ as well. Thus the edge between S and T is either a bi-directed edge or an invisible directed edge in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. Hence we have established (2).

The strategy to complete the proof is to show that $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ can be transformed into a supergraph of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$ via a sequence of equivalence-preserving mark changes (Zhang and Spirtes, 2005; Tian, 2005). By (1) and (2), if $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is not yet a supergraph of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, it is because some bi-directed edges in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$ correspond to directed edges in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. For any such directed edge $P \to Q$ in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ (with $P \leftrightarrow Q$ in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$), (2) implies that $P \to Q$ is invisible. It is then not hard to check that conditions in Lemma 1 of Zhang and Spirtes (2005)[22] hold for $P \to Q$ in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, and thus it can be changed into $P \leftrightarrow Q$ while preserving Markov equivalence. Furthermore, making this change will not make any other such directed edge in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ visible. It follows that $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ can be transformed into a Markov equivalent graph that is a supergraph of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$. (We skip the details as they involve conditions for Markov equivalence we didn't have enough space to cover.)

Denote the supergraph by I. It follows that if there is an m-connecting path between A and B given $\mathbf{C}$ in $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$, the path is also m-connecting in I, the supergraph of $\mathcal{M}_{\mathcal{G}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}}$. Because $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ and I are Markov equivalent, there is also an m-connecting path between A and B given $\mathbf{C}$ in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. ■

22. Here is the Lemma: Let $\mathcal{M}$ be a MAG, and $A \to B$ a directed edge in $\mathcal{M}$. Let $\mathcal{M}'$ be the graph identical to $\mathcal{M}$ except that the edge between A and B is $A \leftrightarrow B$ in $\mathcal{M}'$. (In other words, $\mathcal{M}'$ is the result of simply changing $A \to B$ into $A \leftrightarrow B$ in $\mathcal{M}$.) $\mathcal{M}'$ is a MAG and Markov equivalent to $\mathcal{M}$ if and only if
(t1) there is no directed path from A to B other than $A \to B$ in $\mathcal{M}$;
(t2)] For every $C \to A$ in $\mathcal{M}$, $C \to B$ is also in $\mathcal{M}$; and for every $D \leftrightarrow A$ in $\mathcal{M}$, either $D \to B$ or $D \leftrightarrow B$ is in $\mathcal{M}$; and
(t3) there is no discriminating path for A on which B is the endpoint adjacent to A in $\mathcal{M}$.

Proof of Lemma 16

Proof It is not hard to check that for any two variables $P, Q \in \mathbf{O}$, if P and Q are adjacent in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, then they are adjacent in $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ (though the converse is not necessarily true, because an edge not definitely visible in $\mathcal{P}$ may still be visible in $\mathcal{M}$). Furthermore, when they are adjacent in both $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ and $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$, every non-circle mark on the edge in $\mathcal{P}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$ is "sound" in that the mark also appears in $\mathcal{M}_{\underline{\mathbf{Y}}\overline{\mathbf{X}}}$. The lemma obviously follows. ■

Proof of Lemma 21

Proof Spirtes and Richardson (1996), in proving their Lemma 18, gave a construction of an m-connecting path in $\mathcal{M}$ from a d-connecting path in $\mathcal{G}$. We describe the construction below.[23]

Let p be a minimal d-connecting path between A and B relative to $\mathbf{C}$ in $\mathcal{G}$ that is into A, minimal in the sense that no other d-connecting path between A and B relative to $\mathbf{C}$ that is into A is composed of fewer variables than p is.[24] Construct a sequence of variables in $\mathbf{O}$ in three steps.

Step 1: Form a sequence $\mathbf{T}$ of variables on p as follows. $\mathbf{T}[0] = A$, and $\mathbf{T}[n+1]$ is chosen to be the first vertex after $\mathbf{T}[n]$ on p that is either in $\mathbf{O}$ or a (latent) collider on p, until B is included in $\mathbf{T}$.

Step 2: Form a sequence $\mathbf{S}_0$ of variables in $\mathbf{O}$ of the same length as $\mathbf{T}$, which we assume contains m variables. For each $0 \leq n \leq m-1$, if $\mathbf{T}[n]$ is in $\mathbf{O}$, then $\mathbf{S}_0[n] = \mathbf{T}[n]$; otherwise $\mathbf{T}[n]$ is a (latent) collider on p, which, by the fact that p is d-connecting given $\mathbf{C}$, implies that there is a directed path from $\mathbf{T}[n]$ to a member of $\mathbf{C}$. So in this case, $\mathbf{S}_0[n]$ is chosen to be the first observed variable on a directed path from $\mathbf{T}[n]$ to a member of $\mathbf{C}$.

Step 3: Run the following iterative procedure:

k:=0

Repeat

If in $\mathbf{S}_k$ there is a triple of vertices $\langle \mathbf{S}_k[i-1], \mathbf{S}_k[i], \mathbf{S}_k[i+1] \rangle$ such that (1) there is an inducing path between $\mathbf{S}_k[i-1]$ and $\mathbf{S}_k[i]$ relative to $\mathbf{L}$ in $\mathcal{G}$ that is into $\mathbf{S}_k[i]$; (2) there is an inducing path between $\mathbf{S}_k[i]$ and $\mathbf{S}_k[i+1]$ relative to $\mathbf{L}$ in $\mathcal{G}$ that is into $\mathbf{S}_k[i]$; and (3) $\mathbf{S}_k[i]$ is in $\mathbf{C}$ and is an ancestor of either $\mathbf{S}_k[i-1]$ or $\mathbf{S}_k[i+1]$; then let sequence $\mathbf{S}_{k+1}$ be $\mathbf{S}_k$ with $\mathbf{S}_k[i]$ being removed;

k := k+1

Until there is no such triple of vertices in the sequence $\mathbf{S}_k$.

Let $\mathbf{S}_K$ denote the final outcome of the above three steps. Spirtes and Richardson (1996), in their Lemma 18, showed that $\mathbf{S}_K$ constitutes an m-connecting path between A and B relative to $\mathbf{C}$ in $\mathcal{M}$. We refer the reader to their paper for the detailed proof of this fact. What is left for us to show here is that the path constituted by $\mathbf{S}_K$ in $\mathcal{M}$ is either into A or out of A with an invisible edge.

In other words, we need to show that if the edge between $A = \mathbf{S}_K[0]$ and $\mathbf{S}_K[1]$ in $\mathcal{M}$ is $A \rightarrow \mathbf{S}_K[1]$, then this edge is invisible. Given Lemma 9, it suffices to show that there is an inducing path between

23. Their lemma addresses the more general case in which there may also be selection variables. The construction given here is an adaptation of theirs to fit our case.

24. In Spirtes and Richardson (1996), minimality means more than that the d-connecting path is a shortest one, but for this proof one only need to choose a shortest path.

A and $\mathbf{S}_K[1]$ relative to $\mathbf{L}$ in $\mathcal{G}$ that is into A. This is not hard to show. In fact, we can show by induction that for all $0 \leq k \leq K$, there is in $\mathcal{G}$ an inducing path between A and $\mathbf{S}_k[1]$ relative to $\mathbf{L}$ that is into A.

In the base case, notice that either (i) $\mathbf{S}_0[1]$ is an observed variable on p such that every variable between A and $\mathbf{S}_0[1]$ on p, if any, belongs to $\mathbf{L}$ and is a non-collider on p, or (ii) $\mathbf{S}_0[1]$ is the first observed variable on a directed path d starting from $\mathbf{T}[1]$ such that $\mathbf{T}[1]$ belongs to $\mathbf{L}$, lies on p and every variable between A and $\mathbf{T}[1]$ on p, if any, belongs to $\mathbf{L}$ and is a non-collider on p. In case (i), $p(A, \mathbf{S}_0[1])$ is an inducing path relative to $\mathbf{L}$, which is into A, because p is into A. In case (ii), consider $p(A, \mathbf{T}[1])$ and $d(\mathbf{T}[1], \mathbf{S}_0[1])$. Let W be the variable nearest to A on $p(A, \mathbf{T}[1])$ that is also on $d(\mathbf{T}[1], \mathbf{S}_0[1])$. ($W$ exists because $p(A, \mathbf{T}[1])$ and $d(\mathbf{T}[1], \mathbf{S}_0[1])$ at least intersect at $\mathbf{T}[1]$.) Then it is easy to see that a concatenation of $p(A, W)$ and $d(W, \mathbf{S}_0[1])$ forms an inducing path between A and $\mathbf{S}_0[1]$ relative to $\mathbf{L}$ in $\mathcal{G}$, which is into A because p is into A.

Now the inductive step. Suppose there is in $\mathcal{G}$ an inducing path between A and $\mathbf{S}_k[1]$ relative to $\mathbf{L}$ that is into A. Consider $\mathbf{S}_{k+1}[1]$. If $\mathbf{S}_{k+1}[1] = \mathbf{S}_k[1]$, it is trivial that there is an inducing path between A and $\mathbf{S}_{k+1}[1]$ that is into A. Otherwise, $\mathbf{S}_k[1]$ was removed in forming $\mathbf{S}_{k+1}$. But given the three conditions for removing $\mathbf{S}_k[1]$ in *Step 3* above, we can apply Lemma 32 (together with the inductive hypothesis) to conclude that there is an inducing path between A and $\mathbf{S}_{k+1}[1] = \mathbf{S}_k[2]$ relative to $\mathbf{L}$ in $\mathcal{G}$ that is into A. This concludes our argument. ■

Proof of Lemma 22

Proof This lemma is fairly obvious given Lemma 10. Let u be the path m-connecting A and B given $\mathbf{C}$ in $\mathcal{M}$. Let D (which could be B) be the vertex next to A on u. Construct a DAG $\mathcal{G}$ from $\mathcal{M}$ in the usual way: keep all the directed edges, replacing each bi-directed edge $X \leftrightarrow Y$ with $X \leftarrow L_{XY} \rightarrow Y$. Furthermore, if the edge between A and D is $A \rightarrow D$, it is invisible, so we can add $A \leftarrow L_{AD} \rightarrow D$ to the DAG. Then $\mathcal{G}$ is a DAG represented by $\mathcal{M}$. It is easy to check that there is a d-connecting path in $\mathcal{G}$ between A and B given $\mathbf{C}$ that is into A. ■

Proof of Lemma 23

Proof Note that because A is not an ancestor of any member of $\mathbf{C}$, if there is a path out of A d-connecting A and B given $\mathbf{C}$ in $\mathcal{G}$, the path must be a directed path from A to B. For otherwise there would be a collider on the path that is also a descendant of A, which implies that A is an ancestor of some member of $\mathbf{C}$. The sub-sequence of that path consisting of observed variables then constitutes a directed path from A to B in $\mathcal{M}$, which is of course out of A and also m-connecting given $\mathbf{C}$ in $\mathcal{M}$. The converse is as easy to show. ■

Proof of Lemma 27

Proof A path definitely m-connecting A and B given $\mathbf{C}$ in $\mathcal{P}$ is m-connecting in every MAG represented by $\mathcal{P}$, which is an immediate consequence of the definition of PAG. Let D be the vertex next to A on the definite m-connecting path in $\mathcal{P}$ between A and B given $\mathbf{C}$. All we need to show is that if the edge between A and D is not a definitely visible edge $A \rightarrow D$ in $\mathcal{P}$, then there exists a MAG represented by $\mathcal{P}$ in which the edge between A and D is not a visible edge out of A.

Obviously if the edge in $\mathcal{P}$ is not $A \rightarrow D$, there exists a MAG represented in $\mathcal{P}$ in which the edge is not $A \rightarrow D$, which follows from the fact that $\mathcal{P}$, by definition, displays all edge marks that are shared by all MAGs in the equivalence class.

So we only need to consider the case where the edge in $\mathcal{P}$ is $A \to D$, but it is not definitely visible. Now we need to use a fact proved in Lemma 3.3.4 of Zhang (2006, pp. 80): that we can turn $\mathcal{P}$ into a MAG by first changing every $\circ\!\!\to$ edge in $\mathcal{P}$ into a directed edge $\to$, and then orienting the circle component—the subgraph of $\mathcal{P}$ that consists of $\circ\!\!-\!\!\circ$ edges—into a DAG with no unshielded colliders.[25] Moreover, it is not hard to show, using the result in Meek (1995a), that we can orient the circle component—a chordal graph—into a DAG free of unshielded colliders in which every edge incident to A is oriented out of A.

Let the resulting MAG be $\mathcal{M}$. We show that $A \to D$ is invisible in $\mathcal{M}$. Suppose for the sake of contradiction that it is visible in $\mathcal{M}$. Then there exists in $\mathcal{M}$ a vertex E not adjacent to D such that either $E *\!\!\to A$ or there is a collider path between E and A that is into A and every collider on the path is a parent of D. In the former case, since $A \to D$ is not definitely visible in $\mathcal{P}$, the edge between E and A is not into A in $\mathcal{P}$, but then that edge will not be oriented as into A by our construction of $\mathcal{M}$. So the former case is impossible.

In the latter case, denote the collider path by $\langle E, E_1, ..., E_m, A\rangle$. Obviously every edge on $\langle E_1, ..., E_m, A\rangle$ is bi-directed, and so also occurs in $\mathcal{P}$ (because our construction of $\mathcal{M}$ does not introduce extra bi-directed edges). There are then two cases to consider:

Case 1: The edge between E and E_1 is also into E_1 in $\mathcal{P}$. Then the collider path appears in $\mathcal{P}$. We don't have space to go into the details here, but there is an orientation rule in constructing PAGs that makes use of a construct called "discriminating path" (e.g., Spirtes et al., 1999; Zhang, forthcoming), which would imply that if the collider path appears in $\mathcal{P}$, and every E_i ($1 \le i \le m$) is a parent of D in a representative MAG $\mathcal{M}$, then every E_i is also a parent of D in $\mathcal{P}$. It follows that $A \to D$ is definitely visible in $\mathcal{P}$, a contradiction.

Case 2: The edge between E and E_1 is not into E_1 in $\mathcal{P}$, but is oriented as into E_1 in $\mathcal{M}$. This is possible only if the edge is $E \circ\!\!-\!\!\circ E_1$ in $\mathcal{P}$. But we also have $E_1 \leftrightarrow E_2$ (E_2 could be A) in $\mathcal{P}$, which, by Lemma 3.3.1 in Zhang (2006, pp. 77), implies that $E \leftrightarrow E_2$ is in $\mathcal{P}$. Then $\langle E, E_2, \ldots, A\rangle$ makes $A \to D$ definitely visible in $\mathcal{P}$, which is a contradiction. ■

Proof of Lemma 28

Proof Note that since A does not have a descendant in $\mathbf{C}$, an m-connecting path out of A given $\mathbf{C}$ in $\mathcal{M}$ has to be a directed path from A to B such that every vertex on the path is not in $\mathbf{C}$. Then a shortest such path has to be uncovered,[26] and so will correspond to a definite m-connecting path between A and B given $\mathbf{C}$ in $\mathcal{P}$ (on which every vertex is a definite non-collider). This path is not into A in $\mathcal{P}$ because $\mathcal{P}$ is the PAG for $\mathcal{M}$ in which the path is out of A. ■

Proof of Lemma 29

Proof Let D be the vertex next to A on the definite m-connecting path in $\mathcal{P}$. Since the edge between A and D is not into A in $\mathcal{P}$, there exists a MAG represented by $\mathcal{P}$ in which the edge is out of A (which follows from the definition of PAG). Such a MAG obviously satisfies the lemma. ■

25. A triple of vertices $\langle X, Y, Z\rangle$ in a graph is called an *unshielded* triple if there is an edge between X and Y, an edge between Y and Z, but no edge between X and Z. It is an *unshielded collider* if both the edge between X and Y and the edge between Z and Y are into Y.
26. A path is called *uncovered* if every consecutive triple on the path is unshielded (cf. Footnote 25).

References

R.A. Ali, T. Richardson, and P. Spirtes. Markov equivalence for ancestral graphs. Technical Report 466, Department of Statistics, University of Washington, 2004.

S. Andersson, D. Madigan, and M. Pearlman. A characterization of Markov equivalence classes of acyclic digraphs. *The Annals of Statistics* 25(2):505-541, 1997.

D.M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 87-98, Morgan Kaufmann, 1995.

D.M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research* 3:507-554, 2002.

D. Geiger, T. Verma, and J. Pearl. Identifying independence in Bayesian networks. *Networks 20*, pages 507-534, 1990.

Y. Huang and M. Valtorta. Pearl's calculus of intervention is complete. In *Proceedings of 22nd Conference on Uncertainty in Artificial Intelligence*, pages 217-224, AUAI Press, 2006.

C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 403-411, Morgan Kaufmann, 1995a.

C. Meek. Strong completeness and faithfulness in Bayesian networks, In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 411-418, Morgan Kaufmann, 1995b.

J. Pearl. Causal diagrams for empirical research. *Biometrika* 82:669-710, 1995.

J. Pearl. Graphs, causality and structural equation models. *Sociological Methods and Research* 27:226-284, 1998.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2000.

J.W. Pratt and R. Schlaifer. On the interpretation and observation of laws. *Journal of Econometrics* 39:23-52, 1988.

T. Richardson and P. Spirtes. Ancestral graph Markov models. *The Annals of Statistics* 30(4):962-1030, 2002.

T. Richardson and P. Spirtes. Causal inference via ancestral graph models. In P. Green, N. Hjort, and S. Richardson, editors, *Highly Structured Stochastic Systems*. Oxford University Press, USA, 2003.

J. Robins. A new approach to causal inference in mortality studies with sustained exposure periods—applications to control of the healthy worker survivor effect. *Mathematical Modeling* 7:1393-1512, 1986.

S. Shimizu, P.O. Hoyer, A. Hyvarinen, and A. Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research* 7:2003-30, 2006.

I. Shpitser and J. Pearl. Identification of conditional interventional distributions. In *Proceedings of 22nd Conference on Uncertainty in Artificial Intelligence*, pages 437-444, AUAI Press, 2006.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Springer-Verlag., New York, 1993. (2nd ed., MIT Press, Cambridge, MA, 2000.)

P. Spirtes, C. Meek, and T. Richardson. An algorithm for causal inference in the presence of latent variables and selection bias. In C. Glymour and G.F. Cooper, editors, *Computation, Causation, and Discovery*. MIT Press, Cambridge, MA, 1999.

P. Spirtes and T. Richardson. A polynomial time algorithm for determining DAG equivalence in the presence of latent variables and selection bias. In *Proceedings of the 6th International Workshop on Artificial Intelligence and Statistics*, 1996. URL http://citeseer.ist.psu.edu/spirtes97polynomial.html.

P. Spirtes and T. Verma. Equivalence of causal models with latent variables. Technical Report Phil-36, Department of Philosophy, Carnegie Mellon University, 1992.

J. Tian and J. Pearl. On the identification of causal effects. Technical Report, Department of Computer Science, Iowa State University, 2004.

J. Tian. Generating Markov equivalent maximal ancestral graphs by single edge replacement. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 591-598, AUAI Press, 2005.

C. Winship and L.S. Morgan. The estimation of causal effects from observational data. *Annual Review of Sociology* 25:659-706, 1999.

J. Zhang and P.Spirtes. A transformational characterization of Markov equivalence for directed acyclic graphs with latent variables. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 667-674, AUAI Press, 2005.

J. Zhang. *Causal Inference and Reasoning in Causally Insufficient Systems*. PhD dissertation, Department of Philosophy, Carnegie Mellon University, 2006. URL www.hss.caltech.edu/∼jiji/dissertation.pdf.

J. Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, forthcoming.

H. Zhao, Z. Zheng, and B. Liu. On the Markov equivalence of maximal ancestral graphs. *Science in China (Mathematics)*, 48(4):548-562, 2005.

Journal of Machine Learning Research 9 (2008) 1475-1533 Submitted 5/06; Revised 10/07; Published 7/08

Incremental Identification of Qualitative Models of Biological Systems using Inductive Logic Programming

Ashwin Srinivasan[*] ASHWIN.SRINIVASAN@IN.IBM.COM
IBM India Research Laboratory
4-C, Institutional Area, Vasant Kunj Phase II
New Delhi 110 070, India

Ross D. King RDK@ABER.AC.UK
Department of Computer Science
University of Wales, Aberystwyth
Ceredigion, Wales, UK

Editor: Stefan Wrobel

Abstract

The use of computational models is increasingly expected to play an important role in predicting the behaviour of biological systems. Models are being sought at different scales of biological organisation namely: sub-cellular, cellular, tissue, organ, organism and ecosystem; with a view of identifying how different components are connected together, how they are controlled and how they behave when functioning as a system. Except for very simple biological processes, system identification from first principles can be extremely difficult. This has brought into focus automated techniques for constructing models using data of system behaviour. Such techniques face three principal issues: (1) The model representation language must be rich enough to capture system behaviour; (2) The system identification technique must be powerful enough to identify substantially complex models; and (3) There may not be sufficient data to obtain both the model's structure and precise estimates of all of its parameters. In this paper, we address these issues in the following ways: (1) Models are represented in an expressive subset of first-order logic. Specifically, they are expressed as logic programs; (2) System identification is done using techniques developed in Inductive Logic Programming (ILP). This allows the identification of first-order logic models from data. Specifically, we employ an incremental approach in which increasingly complex models are constructed from simpler ones using snapshots of system behaviour; and (3) We restrict ourselves to "qualitative" models. These are non-parametric: thus, usually less data are required than for identifying parametric quantitative models. A further advantage is that the data need not be precise numerical observations (instead, they are abstractions like positive, negative, zero, increasing, decreasing and so on). We describe incremental construction of qualitative models using a simple physical system and demonstrate its application to identification of models at four scales of biological organisation, namely: (a) a predator-prey model at the ecosystem level; (b) a model for the human lung at the organ level; (c) a model for regulation of glucose by insulin in the human body at the extra-cellular level; and (d) a model for the glycolysis metabolic pathway at the cellular level.

Keywords: ILP, qualitative system identification, biology

∗. Also at: Department of CSE & Centre for Health Informatics, University of New South Wales, Sydney.

1. Introduction

There is a general move in biology from seeking an understanding at the level of individual units (genes, proteins and so on) to an understanding at the system-level. Identifying single genes, proteins or metabolite levels cannot be expected to yield an answer to systemic behaviour any more than a list of radio parts could explain its behaviour (a point made in Lazebnik's humourous and perceptive article: Lazebnik, 2002). What is needed is an understanding of the function of each part and, crucially, how these components are connected, how they are controlled and the dynamic behaviour of the system as a whole. Biology, which for the last decade or so has been pre-occupied with establishing the "parts-list" is now moving to address these other issues. Besides the obvious scientific value of understanding whole systems, substantial benefits are expected to follow in clinical medicine. This is concerned with the application of computation and applied mathematics to improve existing pharmaceutical and medical practices. It is expected that results in systems-level biology will allow a better understanding of the nature of diseases, leading to a targeted design of new drugs and drug treatments. The importance of adopting a systemic approach to biology is not new: there are statements in Darwin's *Origin* that clearly anticipate this need. Its relevance in the modern biological context is summarised in a recent article in *Science* (W.Bialek and D.Botstein, 2004):

> The basic nature and goals of biological research is changing fundamentally. In the past, biological processes and the underlying genes, proteins, other molecules, and environmental factors were of necessity studied one by one in relative isolation. In contrast, today we are no longer satisfied with studies or answers that place each of these in a larger context. We now know that there are tens of thousands of genes encoded in the genome and that simple perturbations such as . . . heat shock, alter the expression of thousands of them . . . New goals are in sight, namely robust mathematical models and computer simulations that faithfully predict the behaviour of entire biological systems.

Some substantial research effort is being expended in trying to achieve these goals. The Physiome Project for example (see `http://www.physiome.org/`) lists its principal aim as being "to understand and describe the human organism, its physiology and pathophysiology quantitatively." This it hopes to achieve by using models at different levels (molecular to organ) that "include everything from diagrammatic schema suggesting relationships to fully quantitative computational models." Similarly, the United Kingdom's main research funding body in biology (the BBSRC) has invested over 15 million pounds in centres for integative systems biology: "the aim is to support research in such a way that all the components of the system under study can be researched at all relevant levels of biological organisation. It necessitates being able to to handle large experimental data sets and having the expertise and capacity to manipulate these and combine them with the theoretical base to develop new predictive and holistic models of how living systems function." (see *Bioscience for Society: A Ten Year Vision, January 2003* at `http://www.bbsrc.ac.uk/about/plans_reports/vision.html`).

In the physical sciences, the principal means of understanding complex systems has been through the use of mathematical models. This same approach is adopted in the field of mathematical biology. Following the pioneering work of Alan Turing (Turing, 1952) and Hodgkin and Huxley (1952) differential equations are now used to model a wide range of transport, reaction and conservation phenomena (Murray, 1993). However, while identification of models of physical systems can often

proceed from first principles (for example, balance equations, energy conservation and so on), the complexity of biological systems often force a much more experimental approach. The modeller selects those physical processes believed to be important, constructs a model and checks if solutions match the observed data. If not, the procedure is repeated until an adequate model is found. For example, a first attempt at modelling oxygen transport to red blood cells may consider a model that accounted for convection, diffusion and chemical reaction (these are the principal physical processes involved). In fact, convection makes a negligible contribution and reaction is only important for sick lungs. Once it is known that only the diffusion term is important, a parametric equation can be found relatively easily.

Broadly speaking, system identification can be viewed as "the field of modelling dynamic systems from experimental data" (Soderstrom and Stoica, 1989). We can distinguish here between: (a) classical system identification techniques, developed by control engineers and econometricians; and (b) machine learning techniques for system identification, developed by computer scientists. While the kinds of models identified by the two kinds of techniques are different, neither provides a foolproof method that can be employed without user interaction.

Classical system identification has concentrated on models largely constrained to be either ordinary differential equations (ODEs) or linear difference equations of some order. With this constraint on model structure, the input-output behaviour of the system is observed over a time interval and some statistical method is used to estimate parameters in the model. In its most general formulation, system identification proceeds by repeated estimation of both structure and parameters until an acceptable model is found. In practice, a small set of structures are given *a priori* and the procedure reduces to one of parameter estimation. Classical techniques have been used to identify linear time-invariant models for purposes of extracting control strategies (in engineering) or time-series predictions (in economics).

In this paper we are concerned instead with using machine learning techniques for system identification. Specifically, our interest is in methods that: (a) are not restricted to specific model structures; and (b) allow the incorporation of domain knowledge both to specify constraints on acceptable model structures and to direct the search through the space of acceptable structures. We believe both these features to be necessary in any empirical approach for identifying biological systems from data. Of the machine learning methods available that are capable of satisfying these requirements, those developed under the framework of Inductive Logic Programming (ILP) are amongst the most powerful. There are two reasons for this. First, the rich logic-based formalism used by ILP methods allows them to represent and identify a wide variety of relational descriptions. Second, ILP methods are unusual in that they make explicit provisions for the incorporation of domain knowledge to guide the model identification process. This includes mechanisms for the requirement in (b) above.

One question that is often raised in the context of ILP is that of efficiency. In the context here, this translates to asking if ILP methods are efficient enough to identify significantly complex biological models? As long as it is reasonable to identify such models in an incremental manner, we believe the answer to this question is "yes" and demonstrate this with the identification of four fairly complex systems at different scales of biological organisation (a predator-prey model at the ecosystem level; a model for the human lung at the organ level; a model for glucose regulation at the extra-cellular level; and the glycolysis metabolic pathway at the cellular level).

A second issue, unrelated to the use of ILP, but relevant to the empirical system identification task is the quantity and quality of data available. The identification of both the structure and parameter of quantitative models (like ODEs) requires a substantial amount of good quality numerical

data. While substantial amounts of quantitative data are being generated at some lower levels of biological organisation (a prominent example is provided by the use of DNA microarray data to estimate mRNA levels in a cell), quality is still variable: it is possible, for example to get very different expression profiles for the same tissue using different microarray technologies (Kuo et al., 2002). At higher levels (for example, at the organ or ecosystem level), data are sparse, although of perhaps better quality. In all cases, we believe it to be substantially easier and more reliable to obtain data that are of a qualitative nature. For example, it may be relatively easier to decide whether certain metabolites are present or absent in a cell, whether their levels have been increasing or decreasing and so on, rather than obtain precise measurements of the metabolites. In this paper, we will be concerned exclusively with system identification from such qualitative data. The resulting models are non-parametric: that is, parameter estimation is not required and data are only needed to identify the model structure. Clearly, these qualitative models cannot be treated as being equivalent to their quantitative counterparts. Nevertheless, they can be used to simulate possible system behaviours and may be much more understandable to a non-mathematical biologist than a quantitative model like a differential equation.

The rest of the paper is organised as follows. Section 2 describes an established approach to qualitative reasoning about dynamic systems. This involves the use of qualitative constraints which form the building blocks of qualitative models (these models include abstractions of ordinary differential equations). Section 3 describes informally the the basics of an ILP system used to identify qualitative models. This includes a variant that performs an incremental identification of increasingly complex models. Section 5.1 demonstrates this form of identification using a model physical system. The application to biological systems is in Section 6. Section 7 examines the automatic identification of stages for the incremental learner. Section 8 concludes the paper. Appendix A provides details of the ILP system used for incremental system identification. Appendix B provides details of the procedure for multi-stage decomposition.

2. Constraint-Based Qualitative Reasoning

Figure 1 (slightly modified from Bratko, 2001) shows four different qualitative abstractions of some numerical statements: (a) numbers are represented by intervals (marked by some distinguished values like *zero*,*end*, *inf* and so on); (b) derivatives are represented by directions of change (like *inc*); (c) functions are represented by monotonic relations (like `MPLUS` denoting "monotonically increasing"); and (d) entire sequences of behaviour are represented by qualitative statements that specify a qualitative values and directions of change.

Reasoning with qualitative abstractions requires a calculus: we propose to use the constraint-based formulation used in the qualitative simulation program QSIM (Kuipers, 1994) (here we provide an informal description along the lines described by Bratko 2001). In this, variables take qualitative values from *domains*. Domains are defined by a name and some ordered set of distinguished values called landmarks. For example, the variable *Amount* in Fig. 1 could be from the domain *level* with landmarks $minf, 0, inf$. A qualitative state of a variable is usually denoted by $Domain : QVal$ where $QVal$ is represented as a $\langle Qmag, Qdir \rangle$ pair, sometimes written $Qmag/Qdir$. $Qdir$ is the qualitative rate of change of the variable, which has a fixed, three-valued resolution (the three quantities being *inc*, for increasing; *dec*, for decreasing; and *std*, for steady). For example, the qualitative state of the variable *Amount* could be $level : 0...inf/inc$ (compare with (d) in Fig. 1).

	Quantitative Statement	Qualitative Statement
(a)	$Level(3.2\ s) = 2.6\ cm$	$Level(t1) = zero...inf$
(b)	$\frac{d}{dt}Level(3.2\ s) = 0.12\ m/s$	$\texttt{DERIV}(Level(t1)) = std$
(c)	$Amount = Level \times Level$	$\texttt{MPLUS}(Level, Amount)$
(d)	Time / Amount: 0.0 / 0.00; 0.1 / 0.02; ... / ...; ... / ...; 159.3 / 62.53	$Amount(zero...end) = zero...inf/inc$

Figure 1: Qualitative abstractions of numerical data (from Bratko, 2001).

The qualitative state of a system is simply a list of the qualitative states of the system's variables and a qualitative behaviour is a list of consecutive qualitative states.

Reasoning is accomplished using constraints. In this approach, pioneered by Kuipers (1994), there are four principal constraints: `ADD(A,B,C)`, for addition of qualitative variables `A` and `B` to give `C`;[1] `MULT(A,B,C)`, to denote $\texttt{A} \times \texttt{B} = \texttt{C}$; `MINUS(A,B)`, for sign inversion $\texttt{A} = -\texttt{B}$; and `DERIV(A,B)`, to denote that `B` is the a derivative of `A`. In this paper, we will also use `SUB(A,B,C)` to denote $\texttt{A} - \texttt{B} = \texttt{C}$. In addition to these, two functional constraints are also used: `MPLUS(A,B)`, to denote that when `A` increases then `B` increases as well; and `MMINUS(A,B)`, to denote that when `A` increases then `B` decreases. We will henceforth refer to these constraints as "the QSIM constraints".

Figure 2 shows a qualitative model for a simple physical system, expressed in terms of the QSIM constraints. A qualitative behaviour of this system—that is, a sequence of qualitative states of the system variables `La`, `Lb` and `Fab` that satisfy the model's constraints—is shown in Fig. 3

A number of advantages have been proposed for using qualitative models. First, in some cases they may be more appropriate than quantitative models. This is particularly so if quantitative measurements are either difficult to obtain or are noisy and what is of interest are the essential properties of the system. Second, the models are quite comprehensible. Both these features are particularly relevant to the modelling of biological systems. There is an additional advantage for automatic system identification of the kind we propose here. Since qualitative models are non-parametric all computational effort is focussed on identifying the model structure. This typically requires data of both less precision and quantity than that required for identification of quantitative models.

1. Constraints apply to the qualitative states of `A`, `B` and `C`. Recall that these are of the form $Domain : Qmag/Qdir$. Thus, the `ADD` constraint ensures that both magnitudes and directions of change are consistent. Thus $\texttt{ADD}(level : 0/inc, level : 0...inf/std, level : inf/inc)$ is true, but $\texttt{ADD}(level : 0...inf/inc, level : inf/std, level : 0...inf/inc)$ is not $(0...inf + inf \neq 0...inf)$. Similarly, $\texttt{ADD}(level : 0...inf/inc, level : 0...inf/std, level : 0...inf/inc)$ is true, but $\texttt{ADD}(level : 0...inf/inc, level : 0...inf/std, level : 0...inf/std)$ is not $(inc + std \neq std)$. It should also be apparent, that while quantitative addition is functional (the sum of a pair of numbers is a unique number) , qualitative addition ones is relational (that is, for a pair of qualitative states for `A` and `B`, `ADD` may be true for more than one qualitative state for `C`). Similar remarks apply to the other constraints.

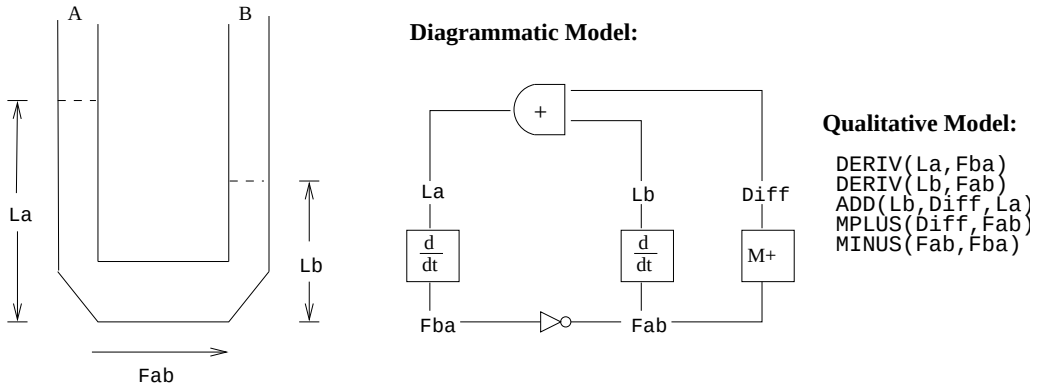


Figure 2: The U-tube and its qualitative model. There are three (measurable) system variables: the water-level in arm A (La); the water-level in arm B (Lb); and the flow of water from A to B (Fab). The diagrammatic model shows the system components involved (two differentiators, an adder, an inverter, and a monotonic function generator) and their interconnections. The qualitative model expresses the same information as a conjunction of constraints (here, we have used the QSIM constraints described in the paper).

La	Lb	Fab
$level : 0/std$	$level : 0/std$	$flow : 0/std$
$level : 0/inc$	$level : 0...inf/dec$	$flow : minf...0/inc$
$level : 0...inf/dec$	$level : 0/inc$	$flow : 0...inf/dec$
$level : 0...inf/dec$	$level : 0...inf/inc$	$flow : 0...inf/dec$
$level : 0...inf/std$	$level : 0...inf/std$	$flow : 0/std$
$level : 0...inf/inc$	$level : 0...inf/dec$	$flow : minf...0/inc$

Figure 3: A qualitative behaviour of the U-tube that is consistent with the qualitative model in Fig. 2. The rows are example states of the qualitative variables and have no implied ordering.

3. Model Identification using Inductive Logic Programming

Given correct definitions for the QSIM constraints, it is our aim in this paper to identify qualitative models such as that shown in Fig. 2, given qualitative states such as those shown in Fig. 3. Since the QSIM constraints are relational, automatic identification of such models clearly requires that the system identification method be able postulate and test relational models. Perhaps the most powerful framework for learning such models is that provided by Inductive Logic Programming (ILP, see Muggleton and Raedt, 1994). ILP is concerned with extracting models in an extremely expressive subset of first-order logic and reasonably efficient implementations have been developed.

To a good first approximation, the basic task addressed by an ILP system can be viewed as a discrete optimisation problem of finding the lowest cost elements amongst a finite set of alternatives. Many ILP systems solve this problem by employing a procedure that searches through a

directed acyclic graph representation of possible models. In this representation, a pair of models are connected in the graph if one can be transformed into another by an operation called "refinement". Figure 4 shows some parts of a graph for the U-tube in which a model is refined to another by the addition of a qualitative constraint. An optimal search procedure (branch-and-bound, for example) traverses this graph in some order, at all times keeping the cost of the best nodes so far. Whenever a node is reached where it is certain that it and all its descendents have a cost higher than that of the best nodes, then the node and its descendents are removed from the search. A portion of the search tree commencing at $\emptyset$ for one such search is shown in Fig. 5.

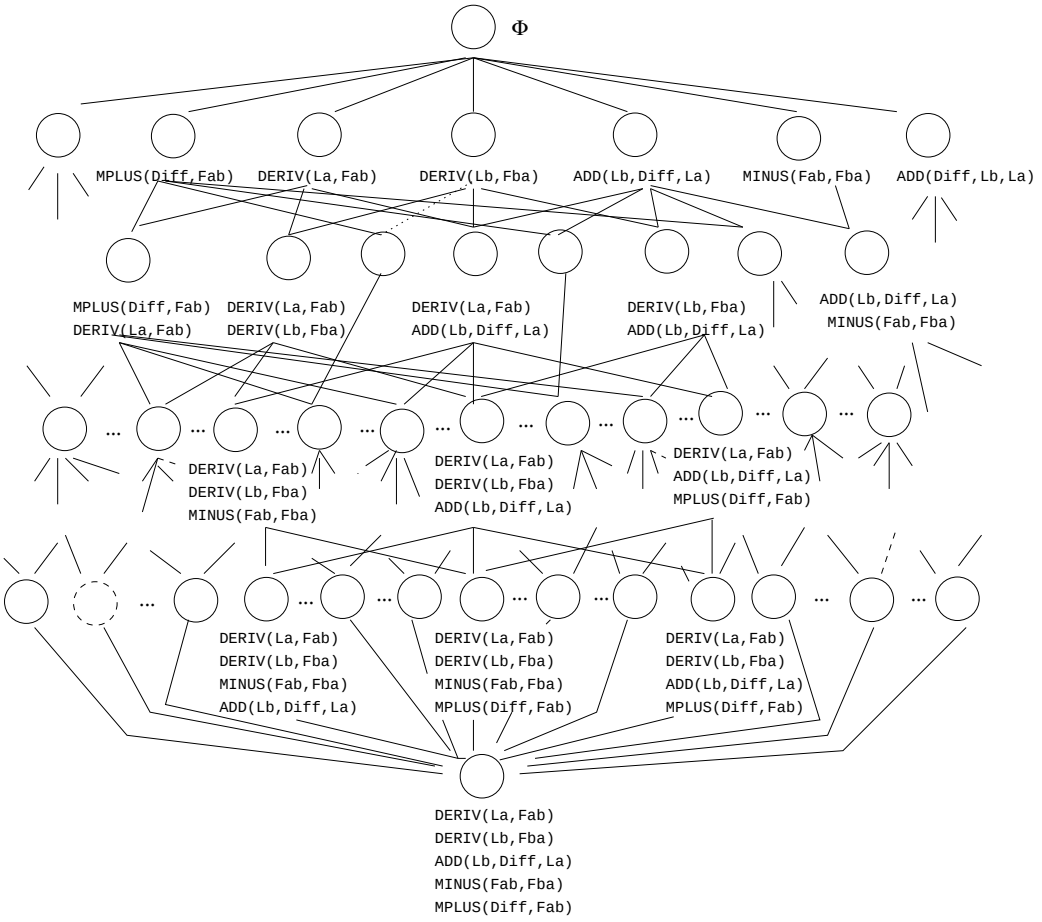


Figure 4: Portions of a refinement graph of models for the U-tube.

Enumerative procedures like branch-and-bound works best if the cost function is monotonic. That is, the score of each node in the search tree is at least as bad as all its descendents (this allows the nodes and its descendents to be removed from the search). The procedure is optimal in the sense that it is guaranteed to find the best solution(s). However in the worst case, it may require examining the entire search space.

Actually, there is more to an ILP system than search. The principal components of such systems are:

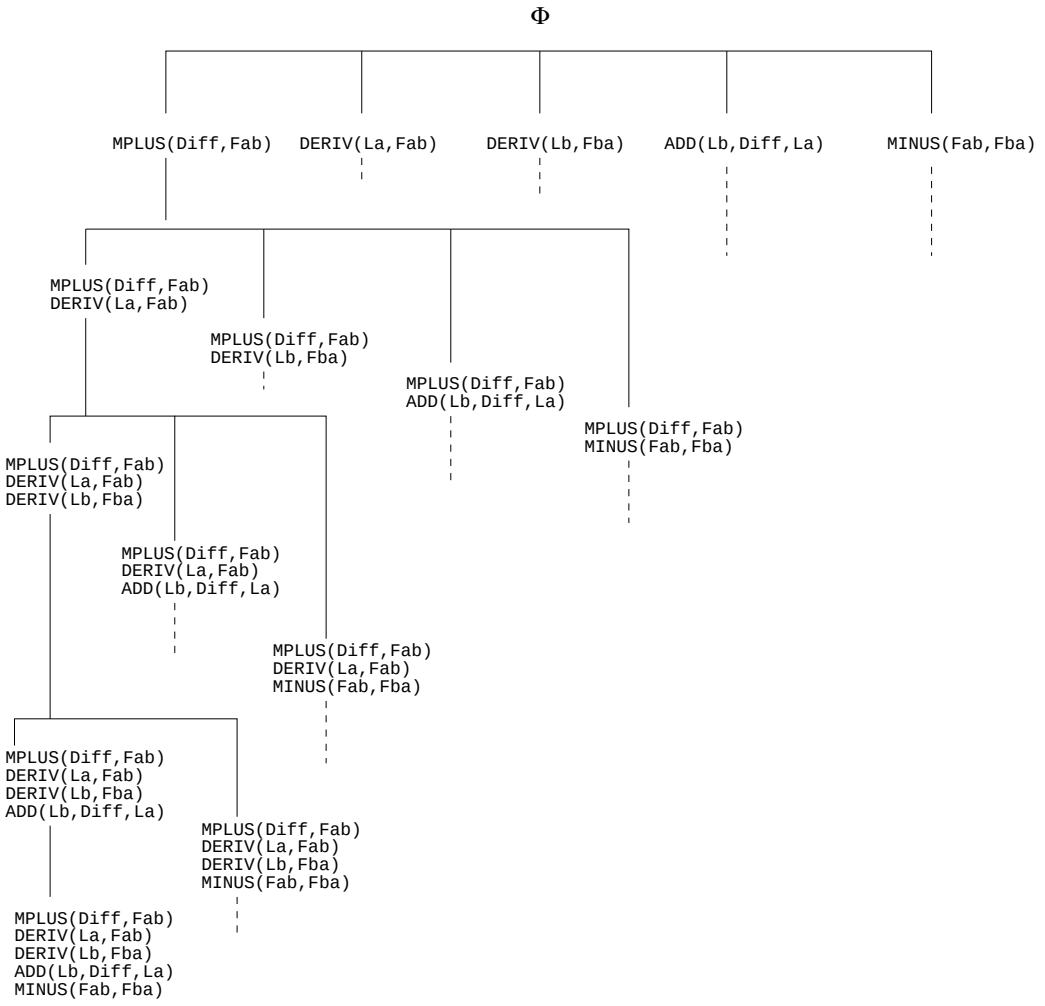


Figure 5: Portions of the search tree explored when searching for models for the U-tube. The search starts from $\emptyset$.

1. *Background knowledge B.* These are statements, usually written in some formal language that specify domain-specific information. We will include in this domain-specific constraints on the kinds of models that are acceptable (or unacceptable, if easier); and directions to the search procedure that allow the system to avoid useless search paths. Examples of these for qualitative model identification are:

   (a) Definitions for qualitative constraints like `DERIV`, `MPLUS`, `ADD` and so on, along with appropriate dimensionality checks *etc.* to ensure their correct usage.

   (b) A constraint specifying that models must not contain relations that are redundant. For example, the relation `ADD(Diff,Lb,La)` is redundant if the model already has `ADD(Lb,Diff,La)` (that is, `ADD` is commutative). The model must respect dimensional

constraints. This prevents relations like `ADD(Lb,Fab,La)` from appearing in the model (`Fab` being a flow has different units of measurement to the level `Lb`).

(c) A directive that the search need not examine models that explain below some proportion of the observed behaviours (more on "explain" in a moment).

2. *Examples E*. These are the observed data. For qualitative model identification, these would be qualitative observations of system behaviour of the form shown in Fig. 3. ILP systems also accept counter-examples of system behaviour. Since this is difficult to obtain for the problems we are concerned with, we do not pursue this further here. Given a set of examples, H is said to explain an observation e if H is consistent with B and e logically follows from B and H (see Appendix A for a precise mathematical formulation). For example, given correct definitions for the qualitative constraints `DERIV`, `MPLUS`, `ADD` and `MINUS` as background knowledge, the qualitative model described by the conjunction `DERIV(La,Fab)` $\wedge$ `DERIV(Lb,Fba)` $\wedge$ `ADD(Lb,Diff,La)` $\wedge$ `MPLUS(Diff,Fab)` $\wedge$ `MINUS(Fab,Fba)` is an explanation of the examples in Fig. 3.

3. *Refinement operator* ρ. This function defines the set of descendents for each node in the refinement graph. With most ILP systems, the set of descendents of a node are (minimal) generalisations or specialisations of the node. Roughly speaking, for qualitative models, generalisations correspond either: to removing one or more qualitative components from the diagrammatic model; or to "disconnecting" qualitative components from each other. Conversely, specialisations correspond to adding new components or connecting existing components.

4. *Cost function* f. This is a real-valued function for each node in the refinement graph. As mentioned earlier, monotonic cost functions are of some importance. A simple cost function satisfying this property in Fig. 4 is $f(H) = -P$, where P is the number of examples explained by model H. If every element H' of $\rho(H)$ contains at least one additional constraint, it can be shown that number of examples explained by H' (and recursively, all its descendents) would be at most P. It follows therefore that the cost of H is no worse than any of its descendents. In practice such a cost function is too simple to be of use (the search would trivially return the most general model), and modifications are made either to: (a) incorporate a trade-off between the explanatory power of the model and its complexity (Muggleton, 1996); or (b) include additional constraints in the background knowledge that prevent the selection of trivial models.

A description of an ILP implementation that uses these components can be found in Section A.2.

4. Identification of Qualitative Models

We refer the reader to Coghill et al. (2005) for an extended review of the literature on learning qualitative models. Briefly, Bratko and colleagues (Bratko et al., 1989; Mozetic, 1987) appear to have the been the first to use qualitative reasoning to build a static model for the electric activity of the heart. Coiera's GENMODEL (Coiera, 1989a,b) was the first machine learning system that constructed qualitative models for dynamic systems. A special-purpose ILP system, GENMODEL (and an updated version in Hau and Coiera, 1997) is restricted to finding qualitative relationships amongst the observed variables only (that is, no intermediate, or hidden, variables are hypothesised).

Model-identification systems that allowed intermediate variables were developed independently by Richards et al. (1992) and Bratko et al. (1992). Both use general-purpose ILP learners (although in different ways) and the principal advantages and shortcomings of these approaches and a later program called QSI (Say and Kuru, 1996) have been documented elsewhere (Coghill et al., 2005).

More recently, the QOPH system for identifying qualitative models exploited the possibility of providing the ILP system ALEPH with a special-purpose refinement operator (Coghill et al., 2002). This operator, with certain "built-in" constraints on acceptable qualitative models, is used by ALEPH to search the space of possible models. Extensive experiments are reported by Coghill et al. (2002) on the reconstruction of some model physical systems. While the results are promising, the scalability of the approach is unclear, since: (a) Model identification is assumed to be possible in a single step. Some simple complexity arguments (see Section A.2) suggest that the complexity of this task grows exponentially with the number of constraints in the model (this is the primary motivation for the incremental approach described in the next section); and (b) The special-purpose refinement operator is difficult to modify and its properties are difficult to analyse. Although not using a general-purpose ILP system, Suc and colleagues have proposed a hybrid approach of combining a logic-based qualitative learner followed by numeric modelling to construct quantitative models of systems (Suc et al., 2003). The approach, called Q^2-learning, first constructs "qualitative model trees". These are like decision trees, with monotonic QSIM constraints in the leaves. These constraints are then used to direct the construction of quantitative models (usually linear models). The success of this approach depends on the availability and quality of numeric data and the system being modelled by a composition of quantitative models (for example, like piecewise linear models).

The advantages of using of a purely qualitative representation for modelling metabolic pathways has been recently advocated (King et al., 2005). In that paper, a special-purpose system is used to generate possible models for the glycolysis pathway. The approach we propose here differs from that in two principal ways. First, it is a general approach as opposed to a specialised one for metabolic pathways. Second, the complexity of the implementation by King and his co-workers is of the same order as QOPH implementation. The incremental approach described in the next section will usually be significantly more efficient.

5. Incremental Model Identification with ILP

Modern ILP systems are largely "one-shot" model constructors. That is, given B, E, ρ and f, they attempt to identify models with the lowest cost in a single search. While this approach has been reasonably successful in the identification of small to medium-sized models (for example, qualitative models containing no more than 4 to 5 constraints), it is unclear whether the approach can scale up to the identification of substantially complex models. For example, the worst-case bound in Remark 2 in Section A.2 grows exponentially in the number of qualitative constraints in the model.

An obvious approach to control this increase in complexity is to decompose system identification into a series of stages, with the final model being some composition of models obtained at each stage. In this paper we use a simple incremental composition in which models identified at any stage are modified at the next stage. That is, one or more models are identified for the first stage (these are all the models consistent with the background knowledge and have the lowest cost). Each of these are then used to give models for the next stage and so on. This is easily achieved by starting the search at each stage at nodes in the refinement graph corresponding to the models found at the

previous stage. Formally the incremental learner is principally provided with: (a) an initial set of models H_0; (b) a sequence of pairs (B_i, E_i) corresponding to the background knowledge and examples for each stage i (the B_i and E_i do not all have to be different); (c) a general-purpose refinement operator ρ for all stages; and (d) a cost function f for all stages. The task of the incremental learner is to construct a set of models from this data (see Fig. 6 and Appendix A).

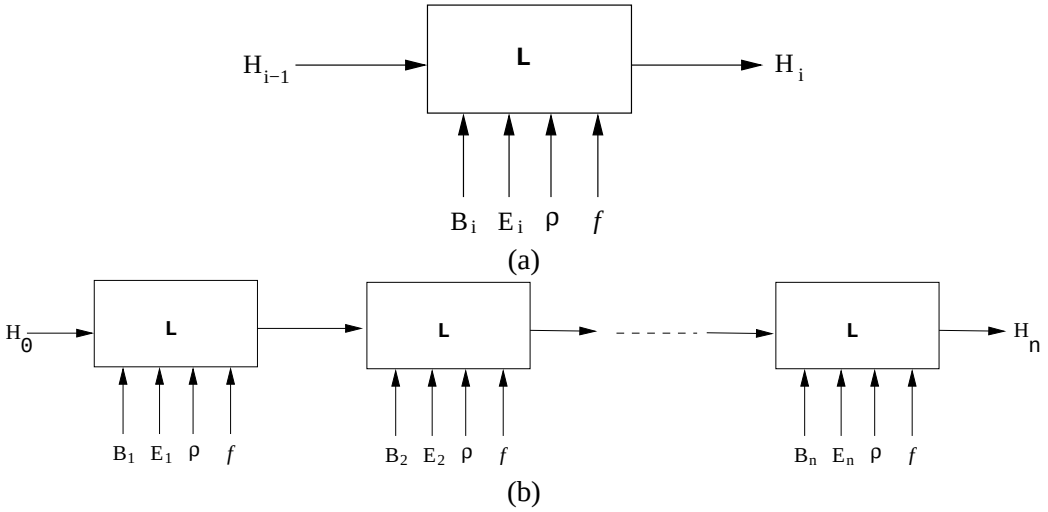


Figure 6: Incremental model identification with ILP. The basic element shown in (a) consists of an ILP learner L that takes as input a set of models, background knowledge, examples, a refinement operator ρ and a cost function f. In (b), this basic unit is repeatedly used to construct a model in n stages. "One-shot" model identification by normal ILP systems is a special case of this process, with $n = 1$ and $H_0 = \{\emptyset\}$ (here $\emptyset$ denotes the empty model).

The actual implementation in Section A.2 contains some additional aspects which are not shown in Fig. 6 (and similar figures in Section 6) for simplicity:

1. A refinement operator that performs both generalisations and specialisations can completely revise models found at a previous stage. However, this is computationally expensive. Instead, we use a refinement operator ρ_A that is restricted to performing specialisations only (for qualitative models, this amounts to adding qualitative constraints and connecting existing qualitative components). To correct partially for this shortcoming, models are subject to a limited generalisation before submission to any L. For qualitative models, this translates to retaining the qualitative components found at the previous stage but disconnecting some or all components, respecting any constraints provided on the usage of the components (in ILP terminology, this amounts to removing variable co-references, respecting any language constraints provided). This allows the incremental procedure to perform a particular kind of revision of models found at the previous stage;

2. Logically redundant models produced by any L are removed and a subset of the result is selected randomly;

3. A cost function f_{Bayes} described in Muggleton (1996) is used. For qualitative models, this performs a trade-off between the likelihood of a qualitative model and its complexity (a quantity related to the number of constraints in the model); and

4. An upper-bound is provided on the amount of search to be conducted by any L.

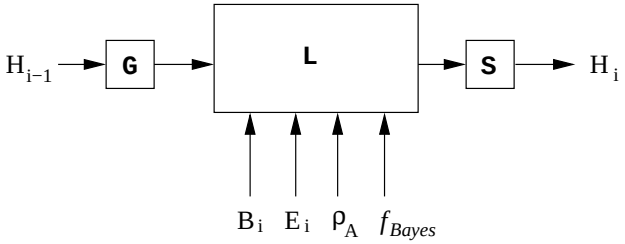


Figure 7: A more accurate representation of the implementation of the basic element used in this paper. Here G performs a limited generalisation of the input models, and can be eliminated for refinement operators that perform both generalisations and specialisations. S performs a random selection of the output models and can be eliminated if all models produced by L are sent to the next stage. ρ_A is a refinement operator that performs specialisations only; and f_{Bayes} is a Bayesian cost function. For simplicity, we will not show G and S in subsequent figures.

A more accurate representation of the basic element of the incremental learner, as implemented by the procedures in Section A.2, is shown in Fig. 7. With these implementation choices it can be shown that, for identification of qualitative models, the size of the search space of such an incremental procedure is dominated by maximum number of additional qualitative constraints that need to be identified at any one of the stages (see Section A.2 for the details). The savings over a non-incremental approach can be substantial, but two points are worthy of re-emphasis:

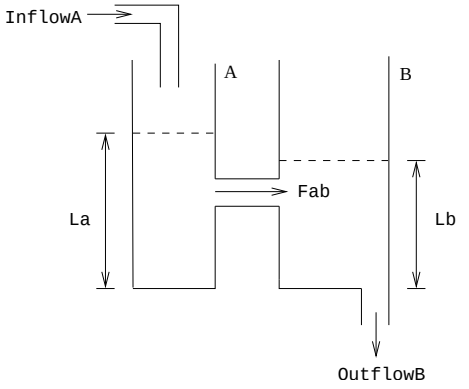
1. The incremental approach requires that a domain-specific decomposition into stages should be possible (by providing background knowledge and observational data for each stage); and

2. We can only guarantee correctness of the incremental approach to the extent that any model identified for a stage will logically entail the observations for that stage (given the background knowledge). The approach cannot, however, provably identify the lowest cost model in the search space. This follows naturally from the fact only lowest cost models are retained at the end of each stage: unless the cost function exhibits a form of monotonocity with stages (or we are simply constructing models in a single-stage), this is tantamount to using a greedy search, which is known to be sub-optimal.

These caveats aside, the incremental approach can be used either: (a) as a "single-shot" model constructor; or (b) to refine approximate models; or (c) to build increasingly larger models using sub-components of smaller ones. In the next section, we illustrate (c) using a model physical system.

5.1 Incremental Qualitative Model Identification: An Example

We consider identifying the qualitative model of the coupled-tanks system shown in Fig. 8. The measurable system variables are these: the input, `InflowA`, that pours into the top of tank A; the output, `OutflowB`, that pours out of the base of tank B; the flow of water from A to B, `Fab`; and the water-levels `La` and `Lb`.



Qualitative Model:

```
DERIV(La,NetflowA)
DERIV(Lb,NetflowB)
ADD(Lb,Diff,La)
MPLUS(Diff,Fab)
MPLUS(Lb,OutflowB)
ADD(NetflowB,OutflowB,Fab
ADD(Fab,NetflowA,InflowA)
```

Figure 8: A system comprised of two coupled tanks and its qualitative model.

The coupled tanks system consists of two tanks connected together. This allows us to decompose the identification of this model into two stages. In the first stage, we focus on identifying a model for tank B, using the single tank system in Fig. 10 (often called the "bathtub" system in qualitative modelling literature). Any consistent models identified for the single tank system are then extended to return final models for the coupled tanks system (see Fig. 9).

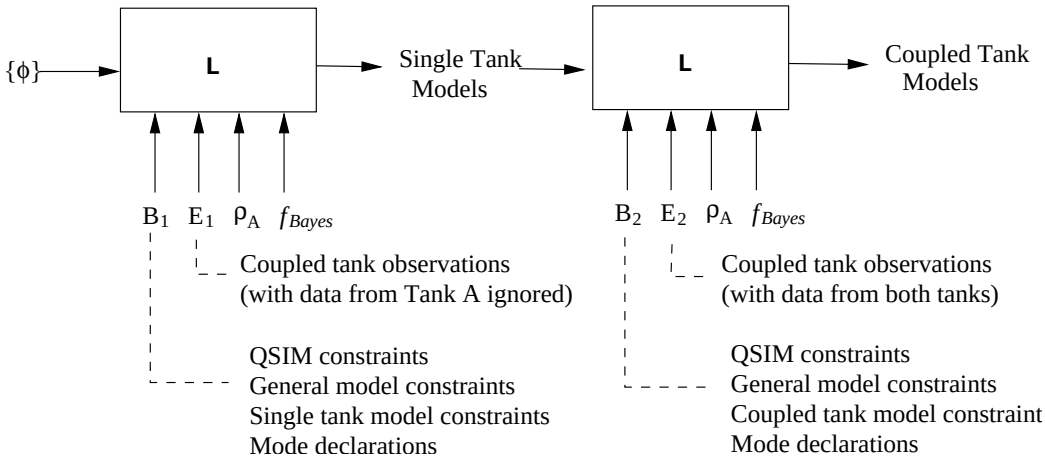
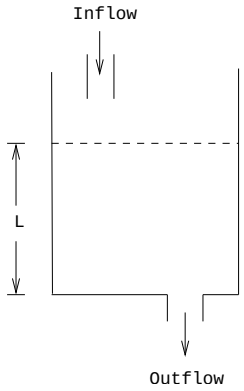


Figure 9: Incremental model identification for the coupled tanks system. Models are first identified by `L` for a single tank system. These are then refined by `L` to models for the coupled tank system. The term "mode declarations" is used in the sense described in Muggleton (1995) and refer to statements that provide domain and connectivity information for the qualitative variables.

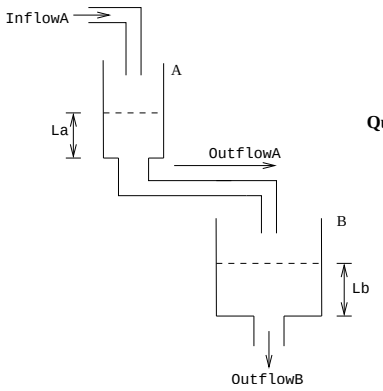
We note in passing that the final model for the coupled tanks is not simply a conjunction of two single tank models. This conjunction would not capture the fact that the flow from tank A to B is related to the difference in levels of fluid in A and B. The conjunction of the two models is, in fact, an appropriate model for the system of cascaded tanks shown in Fig. 11.



Qualitative Model:

```
DERIV(L,Netflow)
MPLUS(L,Outflow)
ADD(Outflow,Netflow,Inflo
```

Figure 10: A single tank system with an input and output. The system variables are `Inflow`, `Outlflow` and , `L`.



Qualitative Model:

```
DERIV(La,NetflowA)
MPLUS(La,Fab)
ADD(OutflowA,NetflowA,InflowA)
DERIV(Lb,NetflowB)
MPLUS(Lb,OutflowB)
ADD(NetflowB,OutflowB,OutflowA
```

Figure 11: A system comprised of two cascaded tanks. The qualitative model is simply a conjunction of two single tank models.

We elaborate further on the elements in Fig. 9:

1. *Background knowledge.* This is comprised of the following components:

   (a) Correct definitions of the QSIM constraints. Our definitions are based on those in Bratko (2001);

   (b) A set of general constraints on "well-posed" qualitative models. We describe these in more detail below;

(c) Stage-specific constraints on the models constructed. This consists of specifying the number of qualitative constraints in the final model for each stage. This is 3 for Stage 1 (the single tank model) and 7 for Stage 2 (the coupled tanks model); and

(d) Stage-specific "mode" declarations similar to the description in Muggleton (1995) that provide domain and connectivity information for the qualitative variables (see Fig. 12);

2. *Examples.* These are in the form of qualitative states for the system variables. Recall that for the coupled tanks system these are: `La`, `Lb`, `InflowA`, `Fab` and `OutflowB` (see Fig. 8). Clearly, flows and levels cannot be negative: we are further only interested in a system with a steady, non-negative input flow. That is, the only valid qualitative state for `InflowA` is $flow:0...inf/std$. `OutflowB`, on the other hand, can be any one of $flow:0/std$, $flow:0/inc$, $flow:0...inf/std$, $flow:0...inf/inc$, $flow:0...inf/dec$. The level of water `La` or `Lb` for the system can similarly assume any of the following qualitative states: $level:0/std$, $level:0/inc$, $level:0...inf/std$, $level:0...inf/inc$, $level:0...inf/dec$. Examples for Stage 1 ignore the values observed for levels and flows for Tank A (that is, `La` and `InflowA` are ignored: this can be easily specified using the mode declarations). Some observations for Stage 1 are shown in Fig. 13. Examples for Stage 2 contain the qualitative states of all the system variables.

3. *Refinement operator and cost function.* These are the operator ρ_A and f_{Bayes} described earlier.

5.2 General Constraints on "Well-posed" Models

In Coghill et al. (2005), the term "well-posed" qualitative models is used to denote those models that satisfy a number of domain-independent constraints. We use the the following constraints from that report:[2]

1. *Size.* A well-posed model must be of a particular size (measured by the number of qualitative constraints).

2. *Completeness.* The model must contain all the measured variables.

3. *Language.* The number of instances of any qualitative constraint in a well-posed model should be below some prescribed number.

4. *Sufficiency.* The model must adequately explain the observed data. By "adequate", we intend to acknowledge here that due to noise in the measurements, not all observations may be logical consequences of the model.[3] The percentage of observations that must be explainable in this sense is a user-defined value.

5. *Redundant.* The model must not contain relations that are redundant. For example, the relation `ADD(Inflow,Outflow,X)` is redundant if the model already has `ADD(Outflow,Inflow,X)`.

2. This list excludes two constraints from the report: the "Determinate" constraint can be effectively enforced by the "Size" constraint. The "Connected" constraint that requires all intermediate variables should appear in at least two qualitative constraints is enforced by the more general "Irrelevant variables" constraint here. All the constraints are assumed to be encoded in the background knowledge for any given stage.
3. Strictly speaking, the model in conjunction with the background knowledge.

Modes:

```
DERIV(+level,-flow)
ADD(+level,+level,-level)      ADD(+level,-level,+level)
ADD(+flow,+flow,-flow)         ADD(+flow,-flow,+flow)
MPLUS(+level,-level)           MPLUS(+level,-flow)
MPLUS(+flow,-flow)             MPLUS(+flow,-level)
MMINUS(+level,-level)          MMINUS(+level,-flow)
MMINUS(+flow,-flow)            MMINUS(+flow,-level)
MINUS(+level,+level)           MINUS(+flow,+flow)
```

A legal model:

```
MPLUS(L,Outflow)
DERIV(L,Netflow)
ADD(Outflow,Netflow,Inflow)
```

Two illegal models:

```
MPLUS(L,Outflow)
DERIV(L,Netflow)
ADD(Outflow,L,Inflow)
```

(ADD cannot add flows to levels)

```
MPLUS(L,Outflow)
ADD(Netflow,Outflow,Inflow)
DERIV(L,Netflow)
```

(ADD needs Netflow to be known)

Figure 12: Example "mode" declarations for the qualitative constraints. For example, the mode declaration `ADD(+level,+level,-level)` states that given values from domain "level" for the the first two arguments, `ADD` computes a value for the third argument (also from domain "level"). This is thus a simple form of dimensionality check. This prevents the ILP system from constructing model M2 (in which a variable from a "flow" domain is added to one from a "level" domain).

Fab	OutflowB	Lb
$flow : 0...inf/std$	$flow : 0/std$	$level : 0/inc$
$flow : 0...inf/std$	$flow : 0...inf/inc$	$level : 0...inf/inc$
$flow : 0...inf/std$	$flow : 0...inf/dec$	$level : 0...inf/dec$
$flow : 0...inf/std$	$flow : 0...inf/std$	$level : 0...inf/std$

Figure 13: Some example observations of the relevant system variables for identification of a single tank model. No ordering is implied amongst these observations.

6. *Contradictory.* The model must not contain relations that are contradictory given other relations present in the model.

7. *Dimensional.* The model must contain relations that respect the dimensionality of the variables involved (this prevents, for example, constraints like `ADD(Inflow,L,...)` from appearing in models for the single-tank system).

8. *Single.* Well-posed models should not contain two or more disjoint sub-models.

9. *Causal.* The model must be causally ordered (Iwasaki and Simon, 1986). In a simple sense, this requires a variable that appears on the right-hand side of a (qualitative) arithmetic constraint should have appeared on the left-hand side of a constraint earlier in the sequence.

The following constraints on the qualitative variables were also used. These are ad-hoc, but were nevertheless found to be extremely effective in constraining the space of possible models:

10. *New variables.* A well-posed model can contain no more than some prescribed number of new, or "hidden", variables. Increasing this number usually increases the value of b in Remark 3 (this is equal to 1 for the single tank model: the hidden variable is `Netflow`).

11. *Irrelevant variables.* Variables in one constraint that are never used by another constraint are taken to be irrelevant. A well-posed model can contain no more than some prescribed number of irrelevant variables (this is equal to 0 for the single tank model).

12. *Distinct variables.* All variables in any constraint are distinct.

13. *Dynamic variables.* Well-posed models must include `DERIV` constraints for any pre-specified "dynamic" variables (these are variables that are known to change with time).

With these inputs, we summarise the results of using incremental model construction to identify a model for the coupled tanks system. Model construction proceeds in two stages. In the first stage, we attempt to identify a single tank model, by ignoring observations for levels and flows in tank A. Figure 14 shows the well-posed models identified by the system. The model with the lowest cost is extended in an attempt to identify a model for the coupled tanks system. Recall that the models selected from Stage 1 are subject to a limited form of generalisation before attempting to identify a model in Stage 2. The result of this generalisation step is shown in Figure 15. Each of these models are extended in Stage 2 to construct final models for the coupled tanks system: the results are in Fig. 16 (the fourth one is the correct model for the system).

Model No.	Model	Cost

1	`MPLUS(Lb,OutflowB)`
	`SUB(Fab,OutflowB,NetflowB)`
	`DERIV(Lb,NetflowB)`
		−9.13

2	`SUB(OutflowB,Fab,NetflowB)`
	`MMINUS(Lb,E)`
	`DERIV(E,NetflowB)`
		−5.37

3	`SUB(Fab,OutflowB,NetflowB)`
	`MPLUS(Lb,E)`
	`DERIV(E,NetflowB)`
		−5.37

4	`SUB(Fab,OutflowB,NetflowB)`
	`DERIV(Lb,E)`
	`MPLUS(NetflowB,E)`
		−5.37

5	`SUB(Fab,OutflowB,NetflowB)`
	`DERIV(Lb,E)`
	`MMINUS(NetflowB,E)`
		−5.37

Figure 14: Well-posed models for the single tank system identified by the first stage of incremental learning. Only the lowest cost model is returned: the rest are shown here for illustrative reasons.

Model No.	Model
1	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB)
2	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,G)
3	MPLUS(Lb,F) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB)
4	MPLUS(Lb,F) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,H)
5	MPLUS(Lb,F) SUB(Fab,F,NetFlowB) DERIV(Lb,NetFlowB)	

Figure 15: Generalisations of the lowest cost model for the single tank system. These models are extended to identify models for the coupled tank system.

Model No.	Model
1	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) ADD(OutflowB,NetflowA,H) ADD(NetflowB,H,InflowA)
2	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) ADD(NetflowB,NetflowA,H) ADD(OutflowB,H,InflowA)
3	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) MPLUS(InflowA,H) MMINUS(InflowA,H)
4	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) MPLUS(Fab,Diff) ADD(Lb,Diff,La)
5	MPLUS(Lb,OutflowB) SUB(Fab,OutflowB,NetflowB) DERIV(Lb,NetflowB) SUB(InflowA,Fab,NetflowA) DERIV(La,NetflowA) MPLUS(NetflowB,Diff) ADD(Lb,Diff,La)	

Figure 16: Well-posed models identified for the coupled tank system. These were obtained by extending the lowest-cost model obtained for the single tank system in Fig. 14 (these are the first three constraints in all the models here). All models shown here have equal (lowest) cost. Model 4 is the target model

The coupled tank system identification task can clearly be decomposed into two stages: the identification of the single tank system consisting of 3 qualitative constraints, followed by its extension by 4 further constraints. It is instructive to illustrate the gain in efficiency from using the incremental procedure. Figure 17 shows the comparative effects of: (1) *No decomposition*. We

attempt to identify all seven constraints in a single stage (this is the "one-shot" approach); and (2) *Correct decomposition*. The single tank model is identified in Stage 1, and then extended to the coupled tank model in Stage 2. This resulted in the models in Figs. 14 and 16.

Figures 16 and 17 illustrate two points we wish to draw attention to about the procedure we have employed, namely:

1. The result may not be a unique model. For the identification of biological systems about which little is known, we do not see this as being a hindrance; and in many cases may even be preferable. New experiments could be proposed to discriminate between the models.

2. Decomposition can significantly increase the efficiency of system identification. No great significance should also be attached to the fact that the correct model is identified even with inappropriate decompositions—recall that the greedy search procedure employed is sub-optimal—although the robustness demonstrated is heartening, since in practice we may not be in a position to know the correct decomposition.

Decomposition	Correct Model Identified?	Time Taken
None	Yes	$>$ 5 days
Correct	Yes	2037 seconds

Figure 17: The effect of decomposition on system identification. The ILP system without decomposition was halted after 5 days of execution.

Finally, while most of the constraints 1–9 on well-posed models are motivated by some well-understood principles underlying qualitative reasoning, the constraints 10–13 on qualitative variables are not. Figure 18 provides some empirical justification for the use of these constraints, by illustrating the proportion of a (uniform) random sample of 10,000 models, all of which satisfy constraints 1–9, but fail these constraints. Based on this proportion constraint 13 has the single strongest effect, followed by 12, 11, and 10.

6. Applications to Biological Systems

In this section we demonstrate the application of the incremental technique described to biological system identification. The demonstrations here serve a dual purpose. First, they are intended to illustrate the ability of a general-purpose ILP system to identify qualitative models for biological systems at significantly different scales of organisation. For this, we have elected to examine modelling problems at the ecosystem, organ, extra-cellular and cellular levels. Second, we intend to demonstrate the ability of the incremental approach proposed to construct models in three different ways: in a single stage without providing any initial model (thus acting as a "one-shot" system identifier); in a single stage by refining an approximate model provided; and in multiple stages. In all cases, the ILP system will use the refinement operator and cost function described in Appendix A. The background knowledge will also largely be the same, consisting of definitions for qualitative constraints. All tasks are of a re-constructive nature and examples are observations generated using

Constraint	Description	Estimate of Proportion of Models Eliminated
10. New variables	New variables = 1	33.07%
	New variables = 2	8.84%
11. Irrelevant variables	Irrelevant variables = 0	73.81%
	Irrelevant variables = 1	33.15%
12. Distinct variables	Distinct variables = true	98.52%
	Distinct variables = false	0.00 %
13. Dynamic variables	Dynamic variables = true	100.00%
	Dynamic variables = false	0.00 %

Figure 18: Estimates of the reduction in the search space by the constraints introduced on qualitative variables. The last column represents the proportion of 10,000 models that satisfy the constraints 1–9 described in Coghill et al. (2005) but fail the corresponding constraint in the second column.

the target qualitative model. The goal in each case is to examine if this target model is amongst the models identified by the ILP system. More details can be found in Section A.3.

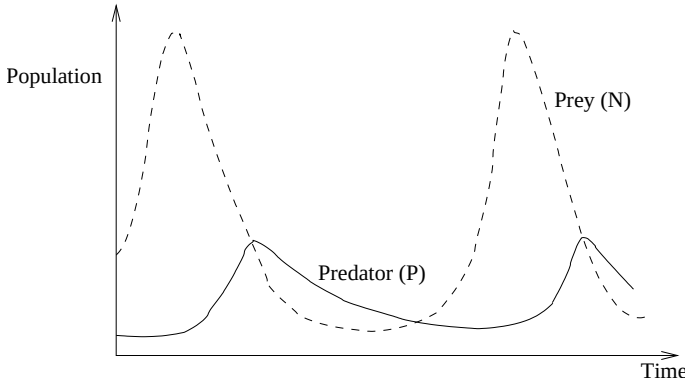
6.1 Ecosystem-Level System Identification

In this section we consider a problem in modelling the dynamics of populations. Specifically we are concerned with identification of a predator-prey model, following the description in Todorovski and Džeroski (2001), which in turn is based on mathematical models developed for the same problem in Murray (1993).

The ecosystem considered is a simple one consisting of populations of predator and prey species—foxes and rabbits, say—that interact in the following manner. Assume that foxes only eat rabbits and that rabbits only eat grass, of which there is an unlimited supply. If the rabbit population is large, the fox population grows. In turn, many rabbits are eaten, resulting in a fall in their numbers. A smaller number of rabbits causes more foxes to die of starvation. Fewer foxes then causes an increase in the rabbit population, which leads to the entire cycle being repeated. This kind of oscillatory behaviour of the two populations is shown in Fig. 19(a). The dynamics of the populations can be modelled using the Lotka-Volterra model, a variant of which is shown in Fig. 19(b). Under certain simplifying assumptions described, the qualitative model is in Fig. 19(c).

We examine reconstructing the model in Fig. 19(c) by using the incremental ILP system as a single-shot model constructor. For this, the ILP system is provided with: (a) the same background knowledge as in Section 5.1; (b) example observations of system behaviour generated using the target model in Fig. 19(c); and (c) the refinement operator and the Bayesian cost function described in Appendix A. The incremental search procedure commences with an empty model (by convention, denoted by $\emptyset$) as the initial hypothesis (see Fig. 20).

The results are shown in Fig.21. Model 1 is the target model. All models were constructed in a single-stage containing 6 qualitative constraints, in approximately 65 seconds of processor time.



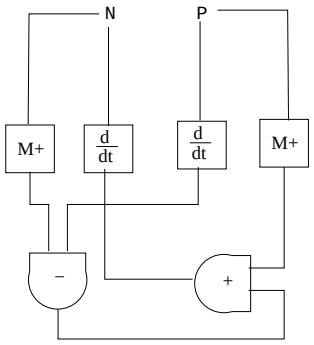
(a)

$$\frac{dN}{dt} = g(N) - c(P)$$

$$\frac{dP}{dt} = c(P) - d(P)$$

(b)

Diagrammatic Model:



Qualitative Model:

```
DERIV(N,Ndot)
DERIV(P,Pdot)
MPLUS(N,G)    ,
MPLUS(P,D)
SUB(G,Pdot,P1)
ADD(P1,D,Ndot)
```

(c)

Figure 19: Modelling predator-prey populations. The changes in populations are shown graphically in (a). There are two system variables: the predator population (P) and the prey population (N). At any given point in time, these variables satisfy the differential equation model in (b). This is a general form of the Lotka-Volterra model for population dynamics. The terms in the model are as follows: $g(N)$ represents the growth-rate of the of the prey in the absence of predators; $c(P)$ is the consumption rate of the predators; and $d(P)$ is the decay-rate of the predators. Under the simple assumptions of $g(N) \propto N$ and $d(P) \propto P$, the corresponding qualitative model is in (c).

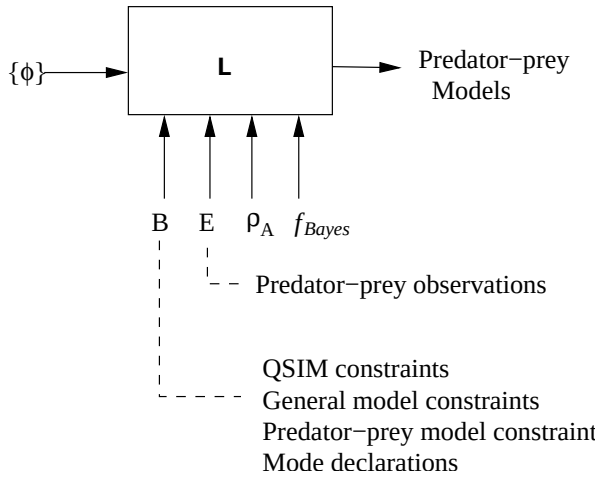


Figure 20: Incremental model identification of predator prey models.

Model No.	Model	Model No.	Model
1	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(P,F) SUB(E,F,G) MPLUS(N,G)	2	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(P,F) SUB(E,F,G) MMINUS(N,G)
3	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(N,F) SUB(E,F,G) MMINUS(P,G)	4	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MMINUS(P,F) SUB(E,F,G) MMINUS(N,G)
5	DERIV(P,Pdot) DERIV(N,Ndot) ADD(Pdot,Ndot,E) MPLUS(N,F) SUB(F,E,G) MINUS(P,G)		

Figure 21: Predator-prey models identified. The target model in Fig. 19(c) is Model 1.

6.2 Organ-Level System Identification

In this section we consider identification of a qualitative model for the human lung. The primary function of the lung is to act as a gas-exchanger. Exchange of gases across a barrier occurs simply because of a difference in pressures. The pulmonary artery carrying blood from the heart contains low concentrations of oxygen and high concentrations of carbon dioxide (at a constant temperature, the concentrations of the gases are proportional to their partial pressures). Oxygen diffuses across the barrier into the blood (and carbon dioxide diffuses into the lung), where is carried by haemoglobin molecules in the pulmonary vein to the heart. This oxygenated blood is then pumped by the heart to the rest of the body using the arterial network. A model of the lung acting in this man-

ner is shown in Fig. 22. The model is constructed using partial pressures of a measurable "marker" gas. A simplification of the model in Fig. 22(c) results from ignoring the blood vessels and treating the lung as a simple gas chamber as shown in Fig. 23(a). The resulting differential equation model is in Fig. 23(b) and the qualitative model is in Fig. 23(c).

We examine reconstructing a model for the lung by providing the ILP system with the approximate model Fig. 23(c): we are interested in investigating whether the ILP system can refine this to the model in Fig. 22(d). The ILP system is provided with: (a) the same background knowledge as in Section 5.1, with additional mode declarations needed for the `MULT` constraint; (b) example observations of system behaviour generated using the target model in Fig. 22(d); (c) the usual refinement operator and cost function. The incremental search is provided with an intial hypothesis consisting of an approximate model for the lung: `MULT(Va,Pa,F)`, `MULT(Vid,Pi,G)`, `DERIV(F,G)` (see Fig. 24).[4]

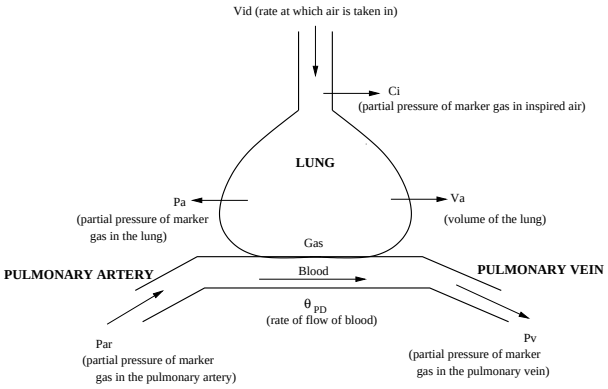
The results, shown in Fig.25, were obtained in 528 seconds of processor time. We note here that model identification required a generalisation of the approximate model provided (`DERIV(F,G)` is changed to `DERIV(F,H)`). Model 2 is the correct model.

6.3 Extra-Cellular System Identification

We use glucose-insulin balance in the human body as a third test case for incremental system identification by ILP. Hormones are chemical messengers, usually small proteins, that play a regulatory role in an organism. Of these, the best known is probably insulin, the first protein whose structure was determined (the amino acid sequence, or primary structure, was determined in 1953 by Sanger and Tuppy). The role of insulin is primarily in maintaining the balance of glucose in the blood. Glucose is used as a source of energy by the central nervous system and by the muscles, and as a source of fat by adipose tissue and the liver, that stores it in the form of a starch called glycogen (see Fig. 26a). If the concentration of glucose in the blood rises too high (usually after digestion of food in the small intestine) then specialised cells in the pancreas are stimulated to produce insulin, by a process involving glycolysis (which we consider in the next section). The presence of insulin signals muscles, fat tissue and the liver to consume glucose, thus lowering it content in the blood. This lower amount of glucose in turn inhibits the production of insulin, and sugar levels rise again until a balance is achieved. This feedback process is not dissimilar to the functioning of a thermostat to maintain a constant temperature in a house. A model of this regulatory mechanism is shown in Fig. 26(b). The model is from Clancy and Kuipers (1994), and is based on a compartmental differential equation model developed by Ironi and Stefanneli (Ironi and Stefanelli, 1994).

Our goal is to reconstruct the qualitative model in Fig. 26(b) using by starting from the empty model. We examine identification of the full model in two stages: the first stage being concerned with identifying the insulin component (the first three constraints in the qualitative model) and the second, the glucose component (the remaining six constraints in the model). As before, the ILP system is equipped with: (a) QSIM relations and their definitions along with additional model-specific constraints; (b) example observatons of system behaviour using the target model in Fig. 26(b); and

4. This is provided *a priori*. In this paper, we do not address how such an hypothesis could have been reached: one possible means could be using the kind of simplified reasoning shown in Fig. 23. There is, of course, nothing preventing the incremental learner described here to start from the empty model $\emptyset$ and construct increasingly better approximations. This would, however, require observational data: something we cannot obtain for the lung model in Fig. 23 (it is impossible to ignore the blood vessels in real-life).



(a)

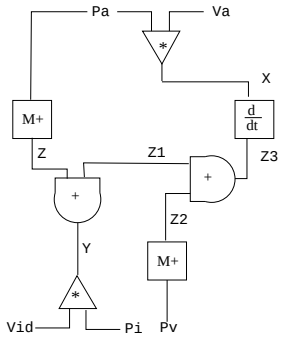
$$\frac{d}{dt}(Va\ Pa) = (Vid\ Pi) - (\lambda\ \theta_{PD}\ (Par - Pv))$$

(b)

$$\frac{d}{dt}(Va\ Pa) = (Vid\ Pi) - (\lambda\ \theta_{PD}\ (Pa - Pv))$$

(c)

Diagrammatic Model:

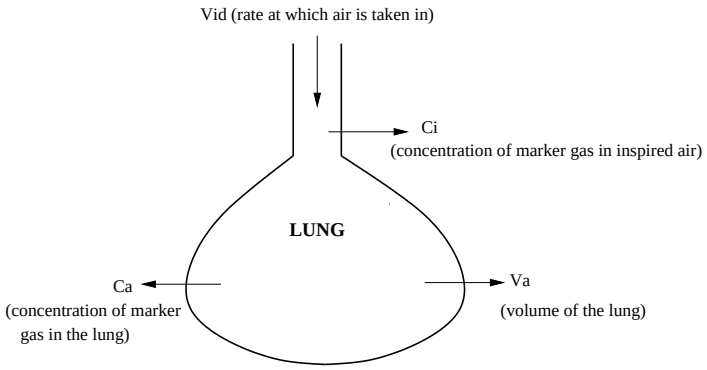


Qualitative Model:

```
MULT(Pa,Va,X)
MULT(Vid,Pi,Y)
MPLUS(Pa,Z)  ,
ADD(Z,Z1,Y)
MPLUS(Pv,Z2)
ADD(Z1,Z2,Z3)
DERIV(X,Z3)
```

(d)

Figure 22: A model for the human lung. In this model, the marker gas is nitrous oxide. There are seven system variables: the rate of inspiration (*Vid*); the concentrations of the marker gas in the inspired air (*Ci*, which is taken to be proportional to the partial pressure *Pi*) and in the lung (*Ca*, proportional to the pressure *Pa*); the volume of the lung cavity (*Va*); the rate of flow of blood θ_{PD}; and the partial pressures in the artery *Par* and the vein *Pv*. On each inspiration, the variables satisfy the differential equation (b). The equation (c) represents the same quantitative model with the assumption that $Pa = Par$, which is reasonable when air is breathed in. The corresponding qualitative model is in (d).

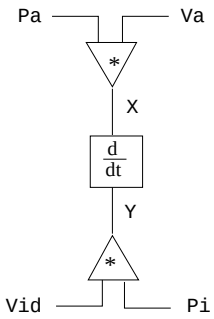


(a)

$$\frac{d}{dt}(Va\ Pa) = (Vid\ Pi)$$

(b)

Diagrammatic Model:



Qualitative Model:

```
MULT(Va,Pa,X)
MULT(Vid,Pi,Y)
DERIV(X,Y)
```

,

(c)

Figure 23: A simplified model of the lung. Ignoring the blood vessels altogether effectively views the lung as the simple gas chamber shown in (a). The resulting quantiative model is in (b) and the corresponding qualitative model is in (c).

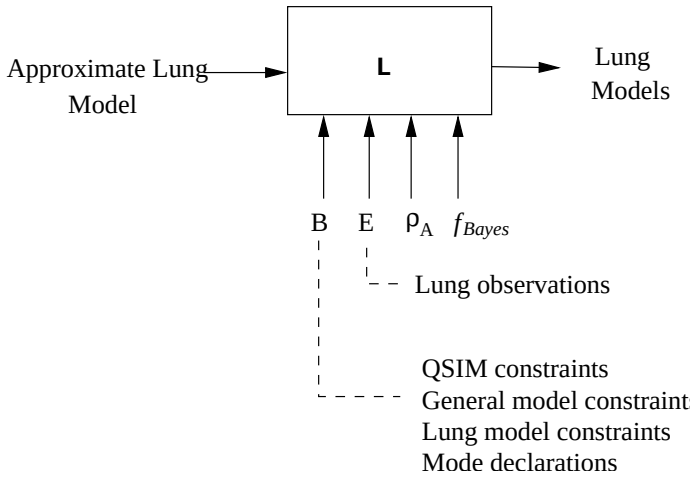


Figure 24: Incremental model identification of lung models.

Model No.	Model	Model No.	Model

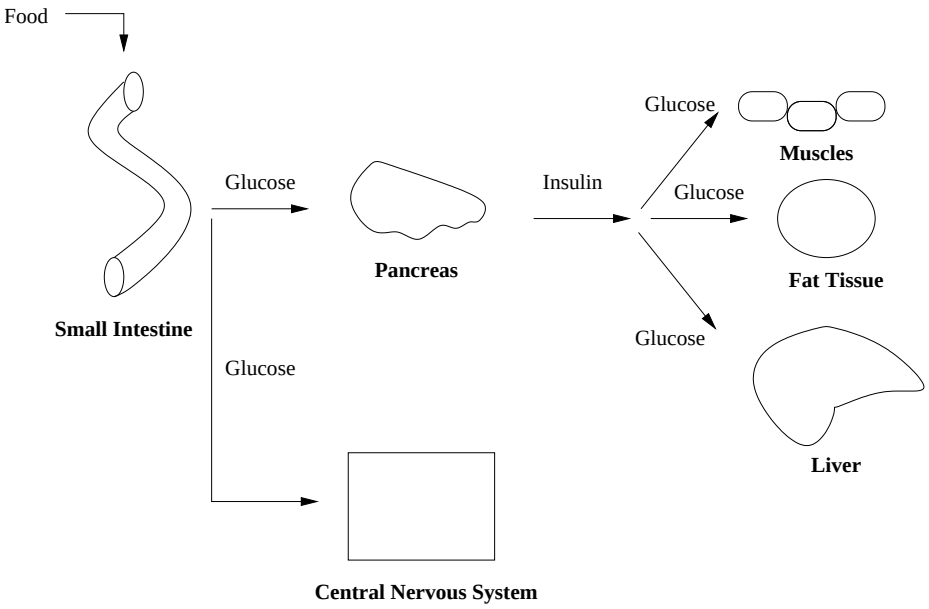
1	`MULT(Va,Pa,F)`
	`MULT(Vid,Pi,G)`
	`DERIV(F,H)`
	`SUB(Pv,Pa,J)`
	`SUB(H,G,I)`
	`MPLUS(I,J)`
		2	`MULT(Va,Pa,F)`
			`MULT(Vid,Pi,G)`
			`DERIV(F,H)`
			`SUB(Pv,Pa,J)`
			`SUB(H,G,I)`
			`MMINUS(I,J)`

3	`MULT(Va,Pa,F)`
	`MULT(Vid,Pi,G)`
	`DERIV(F,H)`
	`SUB(Pv,Pa,I)`
	`SUB(H,I,J)`
	`MPLUS(J,G)`
		4	`MULT(Va,Pa,F)`
			`MULT(Vid,Pi,G)`
			`DERIV(F,H)`
			`SUB(Pv,Pa,I)`
			`SUB(H,I,J)`
			`MMINUS(J,G)`

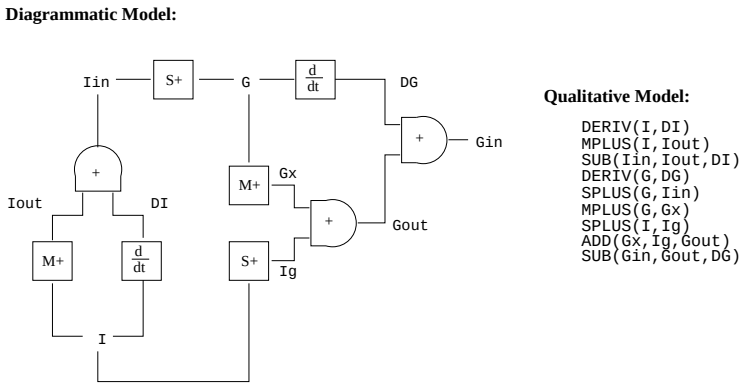
Figure 25: Lung models identified, given the approximate model `MULT(Va,Pa,F)`, `MULT(Vid,Pi,G)`, `DERIV(F,G)`. The target model in Fig. 22(d) is Model 2.

(c) the refinement operator ρ_A and cost function f_{Bayes}. The full system identification process is shown in Fig. 27.

Little difficulty was encountered in identifying the correct constraints for the insulin stage in no more than 1 second of processor time. However, we found it substantially harder to identify the correct constraints for the glucose stage. The principal problems were: (a) a large number of models—over 40, including the one sought—were consistent with the constraints provided; and (b) model evaluation in some cases was extremely slow. We have found two additional constraints to be very useful in reducing the number of models. First, we prevent additions of exogeneous and plasma levels of the same substances (for example, additions of `Iin` and `I`, or `Gin` and `G`). Some plausible justification of this is possible, on the grounds that the two levels are closely related to each other. Second, we prevent monotonic functions of exogeneous inputs `Gin` and `In`, requiring these to be approximated by functions of their counterparts in the blood (that is, `G` and `I`). In addition,



(a)



(b)

Figure 26: Glucose regulation in the blood, shown pictorially in (a), and modelled qualitatively in (b). In the model, `Gin` refers to the glucose intake (in the form of food) and `Iin`, the insulin produced by the pancreas. `G` and `I` are the glucose and insulin levels in the blood. `Gx` is the insulin-independent consumption of glucose by the central nervous system and `Ig` the insulin-dependent consumption of glucose by the muscles, fat tissue and the liver. The qualitative model in Clancy and Kuipers (1994) utilises a sigmoid function `SPLUS`. For the model here, we use the standard `MPLUS` function, which is consistent with the original formulation in Ironi and Stefanelli (1994)

.

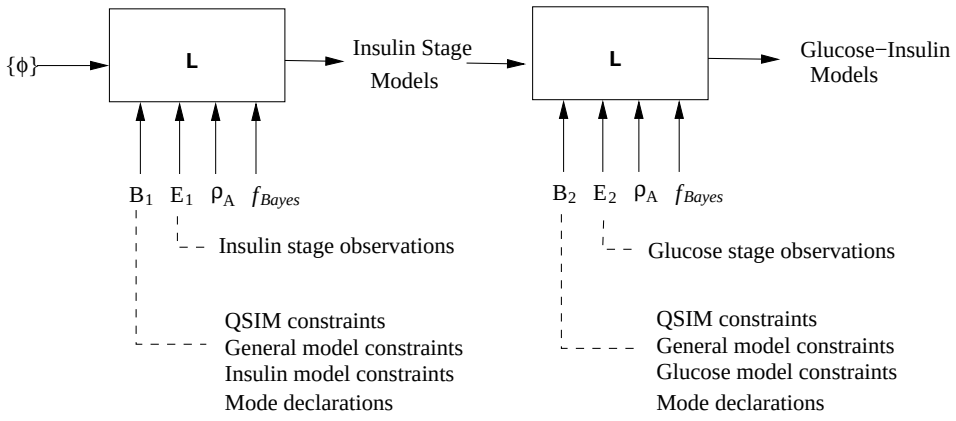


Figure 27: Incremental model identification of models for glucose regulation.

substantially more restrictive mode declarations than in other cases were needed to restrict the search space. Further, we restrict the search to occupy no more than 10,000 seconds of processor time. With these *ad hoc* constraints in place, we are able to repeat model identification using the two stages. The correct insulin model is obtained as before and the results after the glucose stage are shown in Fig.28. Model 3 is equivalent to the target model, given the equivalence of `SPLUS` and `MPLUS` in experiments here.

6.4 Cell-Level System Identification

We use the glycolysis pathway as the final test case for incremental system identification by ILP. Glycolysis is the archetypal pathway. It was historically one of the first to be unravelled, with Otto Meyerhof winning the Nobel prize for discovering key steps in it. Specifically, Meyerhof and colleagues "...were unusually accomplished in breaking down glycolysis into its many separate components, analysing each step separately, then reassembling the constituent parts within an overall system."[5] Glycolysis still presents a challenge to model accurately. The special interest here is that it is significantly different in nature to the models considered so far in the paper, which have all been abstractions of ordinary differential equations. We examine now how the qualitative representation language could be used to develop other kinds of models.

Our qualitative model for glycolysis uses 15 metabolites, namely: pyruvate (pv), glucose (glc), phosphoenolpyruvate (pep), fructose 6-phosphate (f6p), glucose 6-phosphate (g6p), dihydroxyacetone phosphate (dhap), 3-phosphoglycerate (3pg), 1,3-bisphos phoglycerate (1,3bpg), fructose 1,6-biphosphate (f16bp), 2-phosphoglycerate (2pg), glyceraldehyde 3-phosphate (g3p), ADP (adp), ATP (atp), NAD (nad), and NADH (nadh). We have not included H+, H_2O, or Orthophosphate as they are assumed to be ubiquitous. The set of reactions in the pathway are shown in Fig. 29.

We will use the following simple qualitative model for enzymes and metabolites. Metabolites are qualitative variables, whose domains are defined by the name of the metabolite and the landmarks 0 and inf. Qualitative states of the metabolites are restricted to $0/std, 0...inf/std, 0...inf/inc$, $0...inf/dec$. A "qualitative cell-state" is given by the qualitative states of the metabolites of interest in the cell. Enzymes are associated with "qualitative reactions", which result in a qualitative

5. See URL `http://nobelprize.org/physics/articles/states/otto-meyerhof.html`.

Model No.	Model
1	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) ADD(I,Gx,I1) SPLUS(I1,J) SUB(Gin,J,DG)
2	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) ADD(Iout,Gx,I1) SPLUS(I1,J) SUB(Gin,J,DG)
3	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) SPLUS(I,Ig) ADD(Gx,Ig,Gout) SUB(Gin,Gout,DG)
4	DERIV(I,DI) MPLUS(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(G,Gx) MMINUS(I,Ig) ADD(Gin,Ig,J) SUB(J,Gx,DG)
5	DERIV(I,DI) DERIV(I,Iout) SUB(Iin,Iout,DI) DERIV(G,DG) MPLUS(G,Iin) SPLUS(I,Ig) ADD(G,Ig,G1) SPLUS(G1,Gout) SUB(Gin,Gout,DG)	

Figure 28: Models for glucose-insulin regulation. The target model is Model 3.

decrease in the amounts of the reactants and a qualitative increase in the amounts of the products. Examples of each of these are in Fig. 30.

We are interested here in finding a sequence of qualitative reactions that are consistent with the qualitative cell-states before and after glycolysis. For this, we introduce a PATHWAY relation which, for a given sequence of qualitative reactions, holds for pairs of qualitative cell-states $\langle Before, After \rangle$ such that the qualitative state of each metabolite in *Before* can be transformed into its state in $After$ by the qualitative reactions. With this relation, the 3 stage glycolysis process can be modelled as shown in Fig. 31. The reader will note that in this model, reactions proceed sequentially. Of course, biologically speaking, this is not how things happen: reactions that can proceed, do so concurrently. While this can be modelled using a slightly different definition for the PATHWAY relation, the model used here is simpler. There are also good historical reasons to adopt this simpler approach. Glycolsis, as the quote above makes clear, and indeed most other pathways have been uncovered by first experimentally separating them into constituent parts (the qualitative modelling of pathways in (King et al., 2005) did not make this assumption, making the resulting models both difficult to identify—all reactions had to be identified in one-shot—and inefficient to execute).

We examine reconstructing a model for the glycolysis pathway in 3 stages (priming, splitting and phosphorylation). At each stage, the ILP system is provided with: (a) the same background knowledge as in Section 5.1, with additional definitions for the PATHWAY and associated relations. For efficiency, we include three restrictions in the definition of the PATHWAY relation, namely: no

1. (*Hexokinase*): glucose + ATP ⇔ glucose 6-phosphate + ADP.
2. (*Phosphoglucose isomerase*): glucose 6-phosphate ⇔ fructose 6-phosphate.
3. (*Phosphofructokinase*): fructose 6-phosphate + ATP ⇔ fructose 1,6-biphosphate + ADP.

4. (*Aldolase*): fructose 1,6-biphosphate ⇔ dihydroxyacetone phosphate + glyceraldehyde 3-phosphate.
5. (*Triose phosphate isomerase*): dihydroxyacetone phosphate ⇔ glyceraldehyde 3-phosphate.

6. (*Glyceraldehyde 3-phosphate dehydrogenase*): glyceraldehyde 3-phosphate + NAD ⇔ 1,3-bisphosphoglycerate + NADH.
7. (*Phosphoglycerate kinase*): 1,3-bisphosphoglycerate + ADP ⇔ 3-phosphoglycerate + ATP.
8. (*Phosphoglycerate mutase*): 3-phosphoglycerate ⇔ 2-phosglycerate.
9. (*Enolase*): 2-phosphoglycerate ⇔ phospoenolpyruvate.
10. (*Pyruvate kinase*): phospoenolpyruvate + ADP ⇔ pyruvate + ATP.

Figure 29: The reactions comprising the glycolysis pathway. The reactions that consume ATP and NADH are not explicitly included. Glycolysis proceeds in three stages: primary (reactions 1–3), splitting (reactions 4 and 5) and phosphorylation (reactions 6–10). The enzymes involved are in parentheses.

Qualitative states of some metabolites
$atp : 0...inf/std$, $dhap : 0/std$, $nad : 0...inf/dec$

A qualitative cell-state
$\{adp : 0/std, atp : 0...inf/std, f16bp : 0/std, f6p : 0/std, g6p : 0/std, glc : 0...inf/std\}$

A qualitative reaction
$glc + atp \rightsquigarrow g6p + adp$

Some cell-states consistent with $glc + atp \rightsquigarrow g6p + adp$
Before: $\{adp : 0/std, atp : 0...inf/std, f16bp : 0/std, f6p : 0/std, g6p : 0/std, glc : 0...inf/std\}$
After: $\{adp : 0...inf/inc, atp : 0...inf/dec, f16bp : 0/std, f6p : 0/std, g6p : 0...inf/inc, glc : 0...inf/dec\}$
After: $\{adp : 0...inf/inc, atp : 0/std, f16bp : 0/std, f6p : 0/std, g6p : 0...inf/inc, glc : 0...inf/dec\}$

Figure 30: Examples of the qualitative representation used for metabolites, cell-states and chemical reactions. In this, a qualitative reaction causes a qualitative decrease in the reactants and a qualitative increase in the products. The non-determinate nature of qualitative arithmetic means that a cell can be in one of several different states after a reaction.

more than 5 reactions are allowed in a pathway; reactions must use all the metabolites; and reactions have to satisfy some basic constraints of chemical feasibility.[6] In addition, the background knowledge contains an additional constraint that ensures that the model proposed is of the sequential form shown; (b) examples of system behaviour generated using the target model; and (c) the usual refinement operator and cost function. The incremental search procedure commences with the empty model $\emptyset$ as the initial hypothesis (see Fig. 32).

6. The obvious constraint is that products cannot contain elements not available in the reactants. A more sophisticated test estimates the number of chemical bonds broken, and restricts this to at most three: reactions that break more bonds are taken to require an infeasibly large amount of energy, and to be too complex even for an enzyme to manage.

GLYCOLYSIS($Before, After$) if
  PATHWAY($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p, atp + f6p \rightsquigarrow adp + f16bp \rangle$)
  PATHWAY($S_1, S_2, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p \rangle$)
  PATHWAY($S_2, After, \langle g3p + nad \rightsquigarrow 1,3bpg + nadh, 1,3bpg + adp \rightsquigarrow 3pg + atp,$
    $3pg \rightsquigarrow 2pg, 2pg \rightsquigarrow pep, adp + pep \rightsquigarrow atp + pv \rangle$)

where:
PATHWAY($S, F, \langle R_1, R_2, \ldots R_n \rangle$)
  $i = 0$, $S_0 = S$
  for $i = 1 \ldots n$
    QREACTION(S_{i-1}, R_i, S_i)
  $F = S_n$

QREACTION($State, R, NewState$)
  QDECREASE($State, Reactants(R), S$)
  QINCREASE($S, Products(R), NewState$)

Figure 31: A qualitative model for glycolysis. Pathways consist of qualitative reactions, each of which result in a qualitative decrease in the reactants and a qualitative increase in the products. The non-determinacy of qualitative arithmetic means that a qualitative reaction acting on a cell-state could result in one of several new cell-states (since there would be several ways to decrease or increase the qualitative values of metabolites). The system identification task is to find the definition for GLYCOLYSIS given definitions for PATHWAY, QREACTION, QDECREASE and QINCREASE.

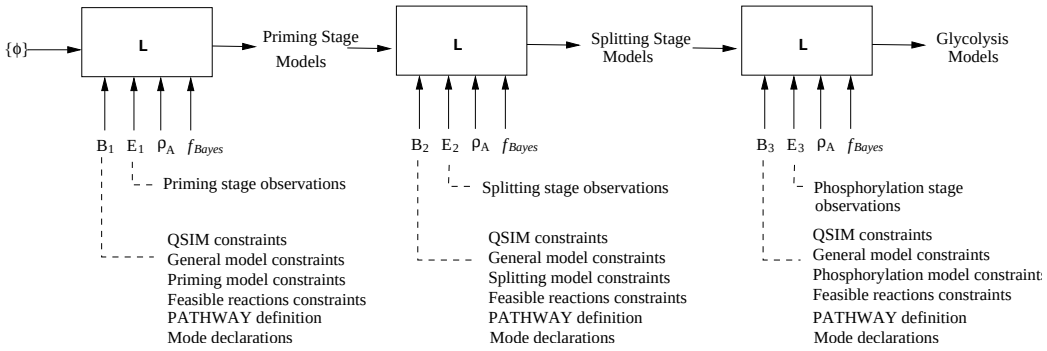


Figure 32: Incremental model identification of models for glycolysis.

The results are shown in Fig. 33. We note here that model identification at Stages 2 and 3 requires a generalisation of the model identified earlier (this removes the co-references to the $After$ variable). The different stages were obtained in 6 seconds (Stage1), 135 seconds (Stage 2) and 5296 seconds (Stage 3).

Stage	Model No.	Model
1	1	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, After, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$)
2	1	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$) PATHWAY($S_1, After, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p\rangle$)
3	1	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$) PATHWAY($S_1, S_2, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p\rangle$) PATHWAY($S_2, After, \langle g3p + nad \rightsquigarrow 1, 3bpg + nadh, 1, 3bpg \rightsquigarrow 3pg,$ $3pg \rightsquigarrow 2pg, 2pg \rightsquigarrow pep, adp + pep \rightsquigarrow atp + pv\rangle$)
	2	GLYCOLYSIS($Before, After$) if PATHWAY(($Before, S_1, \langle atp + glc \rightsquigarrow adp + g6p, g6p \rightsquigarrow f6p,$ $atp + f6p \rightsquigarrow adp + f16bp\rangle$) PATHWAY($S_1, S_2, \langle f16bp \rightsquigarrow dhap + g3p, dhap \rightsquigarrow g3p\rangle$) PATHWAY($S_2, After, \langle g3p + nad \rightsquigarrow 1, 3bpg + nadh, 1, 3bpg + adp \rightsquigarrow 3pg + atp,$ $3pg \rightsquigarrow 2pg, 2pg \rightsquigarrow pep, adp + pep \rightsquigarrow atp + pv\rangle$)

Figure 33: Glycolysis models identified. The target model is Model 2 in Stage 3. The difference in the two models identified in Stage 3 arise in the seventh equation (the second in the last PATHWAY constraint). Model 1 proposes $1, 3bpg \rightsquigarrow 3pg$ and Model 2 proposes $1, 3bpg + adp \rightsquigarrow 3pg + atp$.

7. Decomposition as Search

So far, we have taken the position that the incremental learner will be provided with a decomposition of the system to be identified. While this may be entirely reasonable when we have access to appropriate expertise—a biologist specialising in the kind of systems we are modelling for example—it is of some interest to examine whether a suitable decomposition can be identified automatically. That is, given observations for some system variables, can we automatically decompose the learning task into one that uses a n-stage incremental learner of the form shown in Fig. 6.

Decomposition of complex systems has been studied extensively in econometrics, ever since the pioneering work of Simon and Ando (1961). In this, decomposability of a system is a property of the system by which some subsets of variables (usually non-intersecting) have a greater interaction with each other than other subsets. These subsets define sub-systems into which the larger system can be decomposed. Simon and Ando study the formal properties of linear dynamical systems of the form $x(t+1) = Ax(t)$, where A is some linear operator and the applicability of their results to evolutionary systems has been studied by Shpak et al. (2004a,b). Concerned as we are with a logical representation of a system, our problem is related more to the decomposition of Boolean functions. Most modern work on this stems from that of Ashenhurst (1957) and Curtis (1962). In this, a function f of n variables, denoted here by the set S, is decomposed into Boolean functions h and g, such that $f(X) = h(A, g(B))$, where $A, B \subset S$ and $A \cup B = S$. The techniques are devised for propositional logic, and it is not evident how they could be used to address the decomposition task here. Nevertheless, at least one important principle is directly applicable: both the Ashenhurst and Curtis formulations are essentially procedures that look for suitable decompositions by examining all possible partitions of S. In general though, finding the optimal decomposition of Boolean

functions is computationally hard (see, for example, Boros et al., 1994), and some form of heuristic search is inevitable. This is the basis of the work of Paulson and Wand (1992), who examine the decomposition of a system specified by state variables. Decomposition here means discovering both a partitioning into non-disjoint subsets of system variables, and an assignment of variables in each subset as being "input" or "output". Each subset constitutes a subsystem, and a pair of subsystems are related if an output variable of one is an input variable of another. The final decomposition is the result of a heuristic search process guided by: a set of constraints characterising "good decompositions", a set of rules for enumerating candidate decompositions, and a method of scoring each candidate based on the complexity of the resulting subsystems.

Unlike Paulson and Wand's procedure, we require that related subsystems share models (rather than system variables). Nevertheless, we are able to draw on their basic premise of decomposition being the result of a heuristic search process. Specifically, we use a randomised local search procedure that identifies each stage of the decomposition using a randomised local search procedure that executes the following steps: (a) A subset of system variables is selected randomly from candidate subsets for this decomposition; (b) A model is constructed using this set and its cost determined; (c) All possible "local moves" are constructed. These result in new subsets obtained by adding a system variable not in the original set and by removing a system variable included in the original set; (d) The best local move (the new subset having a model with least cost) is selected and Steps (c)–(d) repeated (the number of repetitions denoted by M). The procedure halts after some fixed number of such iterations, and the best scoring subset is returned. Actually, Steps (a)–(d) are repeated several times (denoted by R) with each repetition starting with a different random subset in Step (a). The best scoring subset across all repetitions is returned and the the entire procedure repeated for the next stage. The reader will recognise the procedure as the GSAT algorithm (Selman et al., 1992) adapted to the problem of automatic decomposition. As with all such procedures, the goal is to obtain an efficient (but sub-optimal) solution to an inherently intractable problem (see Appendix B for details). Needless to say that with this, as with the Paulson and Wand work, automatic decomposition is only worthwhile provided the additional computational burden imposed by searching for the decomposition is less than that of attempting to find the complete model using a one-shot (single-stage) learner.

We apply the procedure to the task of decomposing the coupled tanks system. The reader will recall that this system (shown in Fig 8) is specified by 5 system variables: `InflowA`, `OutflowB`, `Fab`, `La` and `Lb`. The automatic decomposition task is as follows: given values for the 5 system variables, identify the single tank "subsystem" specified by `OutflowB`, `Fab` and `Lb` and then identify the the final model using the single tank submodel and the remaining system variables. Figure 34 summarises the result of employing the randomised local search procedure just described to identify the correct decomposition.

It is evident that for such a small problem, we will quickly explore all of the search space as R and M are increased. Nevertheless, the tabulation shows that very small values of R and M yield variable results. The extent to which the success with moderate values of R and M can be replicated on larger, real systems remains a topic for future research. In Appendix B, we are able to offer some insight by considering artificial problems created by random decompositions of larger sets of variables. These experiments suggest that a 2-stage decomposition of a system like the coupled tanks would require R and M values of approximately 5.

We turn now to the automatic decomposition of the first multi-stage biological system considered in this paper. The glucose-insulin system is comprised of 4 independent system variables

R	M		
	1	2	3
1	0.27	0.70	0.40
2	0.50	0.60	0.90
3	0.40	0.90	1.00

Figure 34: Probability estimates of identifying the correct decomposition of the coupled tanks system, using the randomised local search procedure described in Appendix B. Here R denotes the number of restarts of the randomised procedure and M the number of iterations of local moves. Each entry is is the probability of identifying correctly the single-tank model, followed by the correct coupled tanks model, with the corresponding values of R, M. Probability estimates required for each stage are obtained from 10 repeats of the randomised procedure.

(`Gin`, `G`, `Iin`, `I`), and requires a 2-stage decomposition. Once again, using the experiments on synthetic problems as a guide, we are able to obtain a correct decomposition for this system using low values of R and M (5 in this case). Unfortunately, the decomposition procedure we have just described cannot be used to obtain a decomposition for the glycolysis problem. Here, system variables (metabolites) are re-used across the different stages, which violates a key assumption of the approach (simply speaking, variables used in a stage cannot be re-used at a later stage). This violation makes the search vastly harder, putting in perspective the achievement of Meyerhof and his colleagues.

8. Concluding Remarks

The focus in biology has, until recently, been mainly on individual units. Molecular biology, for example, has mainly focussed on individual molecules and on their properties as isolated entities or as complexes in very simple model systems. However, biological molecules in living systems participate in very complex networks, including regulatory networks for gene expression, intracellular metabolic networks and both intra- and inter-cellular communication networks. Such networks are involved in the maintenance (homeostasis) as well as the differentiation of cellular systems of which we have a very incomplete understanding. Nevertheless, the progress of molecular biology has made possible the detailed description of the components that constitute living systems, notably genes and proteins. Large scale genome sequencing means that we can (at least in principle) delimneate all macromolecular components of a given cellular system. Microarray experiments as well as large scale proteomics will soon give us large amounts of experimental data on gene regulation, molecular interactions and cellular networks. The challenge now becomes to understand how these individual components integrate to complex systems and the function and evolution of these systems, thus scaling up from molecular biology to "systems" biology that provides an understanding at different levels of biological organisation.

Our experience in the physical sciences suggests that the only tractable way of understanding complex systems is through the use of mathematical models. However, biological systems are usually far more complex than physical or human-engineered ones and progress in determining functionality will be crucially dependent on the development of mathematical, and computational tech-

niques specially devised for biological data analysis, modelling and simulation. In this paper, we argue that a qualitative representation of values, along with a powerful machine-learning approach like ILP, provides a useful tool for system-identification at different levels of biological organisation. We have sought to back this claim by demonstrating the use of a general-purpose ILP system to identify models for systems at three disparate levels of biological organisation, namely, ecosystem, organ and cellular. The results are promising, with the target model being amongst a small set of answers returned in each case. While the applications presented have been re-construction of known models, this is clearly not the use we envisage for the approach. Specifically, we expect its principal utility will be in situations where there is some quantitative data of variable quality and quantity, and not much is known about a suitable mathematical model. In these circumstances, the data can be converted to a qualitative representation (in the manner described by Hau and Coiera, 1997) and one or more qualitative models identified. These can then form the basis of understanding the system better and could even be used to direct the construction of quantitative model. For example, they could form the basis of the grammars required for an automated technique such as the one described in Todorovski et al. (2000) (in some sense, this is like extending the Q^2-learning framework in Suc et al. (2003) to the discovery of mathematical models). An additional feature of our work here is that the approach is an incremental one, that seeks to construct the final model in stages. The value of decomposition as an aid to understanding complex systems has long been recognised: Courtois (1985) describes some general principles that motivate the need for such an approach. We believe that when attempting to construct large models with ILP, some form of structured induction (in the sense intended in Shapiro, 1987) would be required. The decomposition into a sequence of stages is an example of such a structuring.

The work presented here has a number of limitations. There are limitations to the power of the qualitative representation used: (1) they can only provide clues to the precise mathematical structure. This may be sufficient for common-sense reasoning about a system, but is clearly insufficient for a complete understanding; (2) simulations with qualitative models can contain spurious behavious; and (3) abstractions appear to be largely restricted to ODE models. It has been suggested that the use of "multivariate constraints" (Wellman, 1991) may allow abstractions of PDE models, but little has been done on that front.

An important limitation of the incremental approach is that the user needs to provide an adequate decomposition of the system-identification task into stages along with the number of constraints in the model for each stage. The latter restriction can be relaxed by providing an upper-bound on the number of constraints. We have described a randomised procedure that attempts to construct a suitable decomposition automatically. The results are encouraging, but the procedure still involves constructing many models and its performance on real problems requires further investigation. The randomised procedure itself is an adaptation of the GSAT algorithm of Selman and colleagues. Minor modifications of this yield procedures akin to WalkSat (Selman et al., 1994) and simulated annealing. Both may yield better algorithms for automatic decomposition than the one here; as would modifications that would allow estimation of model performance without actually requiring their explicit construction.

In the implementation of the incremental learner, the primary limitation of the greedy strategy adopted means that we cannot prove that the models returned are the best possible. A further limitation is the use of a refinement operator that can only perform a restricted kind of refinement of models found at a previous stage: this was done solely to keep the space of possible models within manageable limits (in effect, a limited form of theory-revision is performed). A refinement

operator that performs both generalisations and specialisations could be used, but the computational cost would be substantial.

Finally, applications of the approach have been restricted to re-construction of known target models using simulated data. Clearly, it remains to be shown that similar success can be achieved with real experimental data.

These limitations notwithstanding, we believe the combination of a qualitative representation and an incremental ILP approach to be particularly well-suited to the identifying systems at different levels of biological organisation, for the following reasons: (1) The qualitative representation overcomes some inherent limitations in the data—specifically, noise and sparsity—which make quantitative modelling difficult; (2) Qualitative models provide the correct level of comprehensibility for the non mathematically-minded biologist; and (3) Models of interest usually involve the relationship between a number of different components. Currently, ILP provides the most powerful—and in many cases, the only—framework for identifying such relations, but its use is often hampered by concerns of efficiency. The incremental approach we have described provides one way of overcoming these concerns.

Acknowledgments

The authors would like to thank George Coghill for helping us understand the constraints defining well-posed qualitative models and pointing out an important practical flaw in an early attempt at constructing the lung model. David Gavaghan, of the Computing Laboratory, Oxford helped us with quantitative models for the human lung. Binesh Mangar attempted to construct some of the qualitative models for the coupled tanks and the lung during the course of his M.Sc. at Oxford. Simon Garrett performed extensive experiments with a one-shot ILP model constructor for model physical systems and with a different representation for glycolysis. The results from those experiments were very helpful in motivating the approach described here. Thanks are also due to Ravi Kothari, of the IRL, who drew our attention to the work of Ashenhurst on the decomposition of switching functions. The authors would like to dedicate this paper to the memory of Donald Michie, who died as this paper was being written. He taught us much of what we know in science and machine learning and his guiding hand is greatly missed.

Appendix A. ILP Details

We now describe the specification and implementation details relevant to the ILP system used in the paper.

A.1 Specification

In this paper, we closely follow the specification provided by Muggleton (1994) for an ILP system designed to construct models (usually called hypotheses in the ILP literature) given background knowledge B and observations (usually called examples in the ILP literature) E In this specification an ILP algorithm is one that satisfies the following requirements (reproduced with minor changes from Srinivasan and Kothari, 2005):

Given:

R1. $B \in \mathcal{B}$: background knowledge encoded as statements in logic. This includes I: a set of constraints that should not be violated by an acceptable hypothesis.

R2. $E \in \mathcal{E}$: a finite set of examples $= E^+ \cup E^-$ where:

$E^+ = \{e_1, e_2, \ldots\}$ is a set of definite clauses (these are the positive examples);
$E^- = \{\overline{f_1}, \overline{f_2} \ldots\}$ is an optional set of Horn clauses (these are the negative examples); and
$B \not\models E^+$

Find:

R3. $H \in \mathcal{H}$: a hypothesis such that the following conditions are met:

Sufficiency. This consists of:

$S1$. $B \cup H \models E^+$

Consistency. This consists of:

$C1$. $B \cup H \not\models \square$; and

$C2$. $B \cup H \cup E^- \not\models \square$

The requirement $C1$ ensures that H does not violate any of the constraints I in B. The requirement $C2$ is intended to ensure that H does not contain any over-general clauses. Often, implementations do not require clauses to meet this requirement, as some members of E^- are taken to be noisy. This specification is then refined to allow theories to be inconsistent with some negative examples. We will use the phrase "H explains E, given B" to denote that at least $S1$ and $C1$ are met. An "acceptable H" is any H that explains E, given B.

The specification does not state how acceptable H's are to be constructed, or, if several H's explain the E, then which of them are to be selected. For this, we introduce the following functions:

- A "downward" refinement operator $\rho : \mathcal{H} \rightarrow 2^{\mathcal{H}}$ s.t. $\rho(h) \subseteq \{h' | h \models h'\}$. Given a $h \in \mathcal{H}$, this function returns a subset of the elements of $\mathcal{H}$ that are implied by h.
- A cost function $f : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \Re$. Given a $h \in \mathcal{H}$, $B \in \mathcal{B}$ and $E \in \mathcal{E}$, this function returns an evaluation of h;

Let $\rho^1(h) = \rho(h)$; $\rho^n(h) = \{h'' | \exists h' \in \rho^{n-1}(h)$ s.t. $h'' \in \rho^1_A(h')\}, (n \geq 2)$; and $\rho^*(h) = \rho^1(h) \cup \rho^2(h) \cup \ldots$. With some abuse of notation, let $\rho^1(\{h_1, h_2, \ldots\}) = \rho^1(h_1) \cup \rho^1(h_2) \cup \ldots$; $\rho^n(\{h_1, h_2, \ldots\}) = \rho^n(h_1) \cup \rho^n(h_2) \cup \ldots$; and $\rho^*(\{h_1, h_2, \ldots\}) = \rho^1(\{h_1, h_2, \ldots\}) \cup \rho^2(\{h_1, h_2, \ldots\}) \cup \ldots$

Then, given an initial set of hypotheses $H_0 \subseteq \mathcal{H}$ we specify a particular kind of ILP algorithm $\mathtt{L}(B, E, H_0, \rho, f)$ by modifying the requirement R3 above to:

R3′. $H = \mathtt{L}(B, E, H_0, \rho, f) \subseteq \rho^*(H_0)$: a set of hypotheses such that for each $h \in H$ the following are met:

Sufficiency. This consists of:

$S1$. $B \cup h \models E^+$

Consistency. This consists of:

$C1$. $B \cup h \not\models \square$; and

$C2$. $B \cup h \cup E^- \not\models \square$

Minimal Cost. This consists of:

$F1$. For all $h' \in \rho^*(H_0)$ $f(h', B, E) \geq f(h, B, E)$

We are now in a position to specify an incremental ILP system that uses the algorithm L. Given a finite sequence $\langle S_1, S_2, \ldots, S_k \rangle$ ($k \geq 1$), where each S_i consists of the tuple (B_i, E_i), ($B_i \in \mathcal{B}$ and $E_i \in \mathcal{E}$); $H_0 \subseteq \mathcal{H}$; a downward refinement operator ρ; and a cost function f, find H_k, where $H_i = \mathtt{L}(B_i, E_i, H_{i-1}, \rho, f)$ ($1 \leq i \leq k$).

A.2 Implementation

The basic task of addressed by L described in the previous section can be viewed as a discrete optimisation problem. In general terms, this is posed as follows: given a finite discrete set S and a cost-function $f : S \rightarrow \Re$, find a subset $H \subseteq S$ such that $H = \{s | s \in S \text{ and } f(s) = \min_{s_i \in S} f(s_i)\}$. An optimal algorithm for solving such problems is the "branch-and-bound" algorithm, shown in Fig. 35 (the correctness, complexity and optimality properties of this algorithm can be found in Papadimitriou and Steiglitz, 1982). A specific variant of this algorithm is available within the software environment comprising ALEPH (Srinivasan, 1999). The modified procedure is in Fig. 36. The principal differences from Fig. 35 are:

1. The procedure is given a set of starting points H_0, instead of a single one (i in Fig. 35);
2. A limitation on the number of nodes explored (n in Fig. 36);
3. The use of a boolean function $acceptable : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \{FALSE, TRUE\}$. $acceptable(k, B, E)$ is $TRUE$ if and only if k satisfies requirements $S1$ and $C1$ in Section A.1 (given B and E);
4. Inclusion of background knowledge and examples (B and E in Fig. 36). These are arguments to both the refinement operator ρ (the reason for this will become apparent shortly) and the cost function f.

We now describe an implementation for an incremental procedure for model identification that assumes that the task has been decomposed into a finite sequence of stages $\langle S_1, S_2, \ldots, S_k \rangle$ ($k \geq 1$). Each S_i consists of the tuple (B_i, E_i), where B_i and E_i refer to the background knowledge and examples relevant to stage i. With this decomposition in place, Fig. 37 shows a simple greedy implementation used to identify the final models.
Finally, we turn to some points concerning the implementation used in this paper:

- Qualitative models are represented as definite clauses. Given a definite clause C, the qualitative constraints in the model (the size of the model) are obtained by counting the number of qualitative constraints in C. This will also be called the "size of C".
- Constraints, such as the restrictions to well-posed models, are assumed to be encoded in the background knowledge;
- $acceptable(C, B, E)$ is $TRUE$ for any qualitative model C that is consistent with the constraints in B, given E.

$bb(i, \rho, f)$: Given an initial element i from a discrete set S; a successor function $\rho : S \rightarrow 2^S$; and a cost function $f : S \rightarrow \Re$, return $H \subseteq S$ such that H contains the set of cost-minimal models. That is for all $h_{i,j} \in H$, $f(h_i) = f(h_j) = f_{min}$ and for all $s' \in S \backslash H$ $f(s') > f_{min}$.

1. *Active* := $\langle (i, -\infty) \rangle$.
2. *worst* := ∞
3. *selected* := $\emptyset$
4. while *Active* $\neq \langle\rangle$
5. begin
   (a) remove element $(k, cost_k)$ from *Active*
   (b) if $cost_k < worst$
   (c) begin
      i. $worst := cost_k$
      ii. $selected := \{k\}$
      iii. let $Prune_1 \subseteq Active$ s.t. for each $j \in Prune_1$, $\underline{f}(j) > worst$ where $\underline{f}(j)$ is the lowest cost possible from j or its successors
      iv. remove elements of $Prune_1$ from *Active*
   (d) end
   (e) elseif $cost_k = worst$
      i. $selected := selected \cup \{k\}$
   (f) $Branch := \rho(k)$
   (g) let $Prune_2 \subseteq Branch$ s.t. for each $j \in Prune_2$, $f_{min}(j) > best$ where $f_{min}(j)$ is the lowest cost possible from j or its successors
   (h) $Bound := Branch \backslash Prune_2$
   (i) for $x \in Bound$
      i. add $(x, f(x))$ to *Active*
6. end
7. return *selected*

Figure 35: A basic branch-and-bound algorithm. The type of *Active* determines specialised variants: if *Active* is a stack (elements are added and removed from the front) then depth-first branch-and-bound results; if *Active* is a queue (elements added to the end and removed from the front) then breadth-first branch-and-bound results; if *Active* is a prioritised queue then best-first branch-and-bound results.

- *Active* is a prioritised queue sorted by f;
- The successor function used is ρ_A. This is defined as follows. Let S be the size of an acceptable model and C be a qualitative model of size S' with $n = S - S'$. We assume B constains a set of mode declarations in the form described in Muggleton (1995). Then, given a definite clause C, obtain a definite $C' \in \rho_A(C, B, E)$ where $\rho_A = \rho_A^n = \langle D' | \exists D \in \rho_A^{n-1}(C, B, E)$ s.t. $D' \in \rho_A^1(D, B, E) \rangle, (n \geq 2)$. $C' \in \rho_A^1(C, B, E)$ is obtained by adding a literal L to C, such that:
  - Each argument with mode $+t$ in L is substituted with any input variable of type t that appears in the positive literal in C or with any variable of type t that occurs in a negative literal in C;
  - Each argument with mode $-t$ in L is substituted with of any variable of C of type t that appears before that argument or by a new variable of type t;

$bb_A(B,E,H_0,\rho,f,n)$: Given background knowledge $B \in \mathcal{B}$; examples $E \in \mathcal{E}$; a set of initial elements H_0 from a discrete set of possible hypotheses $\mathcal{H}$; a successor function $\rho : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow 2^{\mathcal{H}}$; a cost function $f : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \Re$; and a maximum number of nodes $n \in \mathcal{N}$ ($n \geq 0$) to be explored, return $H \subseteq \mathcal{H}$ such that H contains the set of cost-minimal models of the models explored.

1. $Active = \langle\rangle$
2. for $i \in H_0$
   (a) add $(i, -\infty)$ to $Active$
3. $worst := \infty$
4. $selected := \emptyset$
5. $explored := 0$
6. while ($explored < n$ and $Active \neq \langle\rangle$)
7. begin
   (a) remove element $(k, cost_k)$ from $Active$
   (b) increment $explored$
   (c) if $acceptable(k,B,E)$
   (d) begin
      i. if $cost_k < worst$
      ii. begin
         A. $worst := cost$
         B. $selected := \{k\}$
         C. let $Prune_1 \subseteq Active$ s.t. for each $j \in Prune_1$, $\underline{f}(j,B,E) > worst$ where $\underline{f}(j,B,E)$ is the lowest cost possible from j or its successors
         D. remove elements of $Prune_1$ from $Active$
      iii. end
      iv. elseif $cost_k = worst$
         A. $selected := selected \cup \{k\}$
   (e) end
   (f) $Branch := \rho(k,B,E)$
   (g) let $Prune_2 \subseteq Branch$ s.t. for each $j \in Prune_2$, $\underline{f}(j,B,E) > worst$ where $\underline{f}(j,B,E)$ is the lowest cost possible from j or its successors
   (h) $Bound := Branch \backslash Prune_2$
   (i) for $x \in Bound$
      i. add $(x, f(x,B,E))$ to $Active$
8. end
9. return $selected$

Figure 36: A variant of the basic branch-and-bound algorithm, implemented within the ALEPH system. Here $\mathcal{B}$ and $\mathcal{E}$ are sets of logic programs; and $\mathcal{N}$ the set of natural numbers.

– Each argument with mode $\#t$ in L is substituted with a ground term of type t. This assumes the availability of a generator of elements of the Herbrand universe of terms; and
– $acceptable(C',B,E)$ is $TRUE$.

The following properties of ρ_A^1 (and, in turn to ρ_A) can be shown to hold (Riguzzi, 2005):

– It is locally finite. That is, $\rho_A^1(C,B,E)$ is finite and computable (assuming the constraints in B are computable);

incsearch(S, H_I, ρ, f, n, m) : Given a sequence of stages $S = \langle (B_1, E_1), (B_2, E_2), \ldots, (B_k, E_k) \rangle$, $(1 \leq k < \infty)$ where $B_i \in \mathcal{B}$; $E_i \in \mathcal{E}$; a set of initial elements H_I from a discrete set of possible hypotheses $\mathcal{H}$; a successor function $\rho : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow 2^{\mathcal{H}}$; and a cost function $f : \mathcal{H} \times \mathcal{B} \times \mathcal{E} \rightarrow \Re$; a maximum number of nodes $n \in \mathcal{N}$ $(n \geq 0)$ to be explored at each stage; and a maximum number of models $m \in \mathcal{N}$ $(m \geq 0)$ to be returned at each stage; and return $H \subseteq \mathcal{H}$

1. $H_0 := randomselect(m, I)$
2. $i := 1$
3. while $(i \leq k)$
4. begin
   (a) $H'_{i-1} := \{h' | h \in H_{i-1}$ and $h' = generalise(h)\}$
   (b) $H'_i := \{h' | h' = bb_A(B_i, E_i, H'_{i-1}, \rho, f, n)\}$
   (c) $H''_i := nonredundant(B_i, H'_i)$
   (d) $H_i := randomselect(m, H''_i)$
   (e) increment i
5. end
6. return H_k

Figure 37: A simple incremental procedure for system identification. Given a decomposition into k stages, the best models found at each stage are refined further.

- It is weakly complete. That is, any clause containing n literals can be obtained in n refinement steps from the empty clause;
- It is not proper. That is, C' can be equivalent to C;
- It is not optimal. That is, C' can be obtained multiply by refining different clauses.

In addition, it is clear by definition that given a qualitative model C, $accep\ table(C', B, E)$ is $TRUE$ for any model $C' \in \rho^1_A(C, B, E)$. In turn, it follows that $acceptable(C', B, E)$ is $TRUE$ for any $C' \in \rho_A(C, B, E)$.

- The cost function used is $f_{Bayes}(C, B, E) = -\mathrm{P}(C|B, E)$ where $\mathrm{P}(C|B, E)$ is the Bayesian posterior probability estimate of clause C, given background knowledge B and positive examples E. Finding the model with the maximal posterior probability (that is, lowest cost) involves maximising the function (McCreath, 1999):

$$Q(C) = \log D_{\mathcal{H}}(C) + p \log \frac{1}{g(C)}$$

where $D_{\mathcal{H}}$ is a prior probability measure over the space of possible models; $p = |E|$, the number of positive examples; and g is the generality of a model. We use the approach used in the the ILP system C-Progol to obtain values for these two functions. That is, the prior probability is related to the complexity of models (more complex models are taken to be less probable, *a priori*); and the generality of a model is estimated using the number of random examples entailed by the model (the details of this are in Muggleton, 1996);

- The function $randomselect(m, H)$ in Fig. 37 randomly selects (without replacement) m elements of the set H (or all the elements of H if its cardinality is less than m);

- For all stages i in Fig. 37, the bb_A constructs no more than n models for each stage. Here we restrict n to 1000;

- For all stages i in Fig. 37, no more than m of the lowest-cost models are returned; Here we restrict m to 1000;

- The function *generalise* in Fig. 37 is restricted to "splitting" variable co-references apart (see Definition 27 and Lemma 31 in Muggleton (1995) and Remark 4 below for more on this); and

- The function *nonredundant* in Fig. 37 returns a set of non-redundant models. Given background knowledge B and a set of models S encoded as definite clauses, a model $C_1 \in S$ is redundant, iff for $S_1 = S - \{C_1\}$, $B \cup S \equiv B \cup S_1$. It can be shown that this entails checking that $B \cup S_1 \models C_1$. $nonredundant(B,H)$ returns all elements $C \in H$ which do not satisfy this redundancy check.

We now report on some properties of the various procedures described. It is evident that *incsearch* in Fig. 37 performs the same function as a non-incremental (single-shot) ILP system if $k = 1$, $H_I = \{\emptyset\}$ (that is, H_I consists of the empty model) and $m \geq n$.

Remark 1 Termination, correctness and sub-optimality *Termination of bb_A follows trivially if the number of nodes searched (n) is finite; and calls to acceptable and f terminate. It is also easy to see that the conditional statement on Step 7c ensures that, for all models $k \in selected$, $acceptable(k,B,E)$ is TRUE. All models returned by bb_A are correct in this sense. Since models returned by incsearch on any iteration i are a subset of the models returned by bb_A, it follows that all models returned by incsearch are also correct. The branch-and-bound procedure is known to be optimal, in that can identify the lowest cost models in the search space $\mathcal{H}$. However, bb_A with $\rho = \rho_A$ is optimal if and only if $n \geq |\mathcal{H}|$ and $H_I = \{\emptyset\}$. It follows that incsearch with $\rho = \rho_A$ is only optimal if and only if $k = 1$, $H_I = \{\emptyset\}$, $n \geq |\mathcal{H}|$, and $m \geq n$.*

Although a general statements about search complexity can be made, the following remarks refer specifically to the search for qualitative models.

Remark 2 Search space for qualitative models. *Let the number of qualitative constraints in acceptable models be restricted to some size d. Given element a single starting element i size d_i, the task of bb_A (we will assume n to be large) is to return all models of size d. This is done by examining all models returned by ρ_A that adds $d - d_i$ constraints to i. In the worst case, each $i = \emptyset$ and ρ_A has to return all models of size d. If the maximum recall number of any mode declaration be bound by some constant b, then there are at most b extensions of size 1, b^2 extensions of size 2 and so on, up to b^d models of size d. That is, given a model i, the number of acceptable models of size d constructed by ρ_A is at most b^d.*

We now consider an incremental procedure that simply selects some of the best qualitative models found at a stage for refinement at the next stage. It follows that the size of the search space depends principally on the maximum number of qualitative constraints added at any stage.

Remark 3 Incremental search space for qualitative models. (simple case). *Assume as before that the target model is restricted to d constraints and the maximum recall of any mode declaration is b. Assume further that model identification can be decomposed into $k \geq 1$ stages, with each stage resulting in models with $d_1, d_2, \ldots, d_k$ constraints (we will assume that all models in the initial set H_I have $d_0 \geq 0$ constraints and that $d_{i+1} \geq d_i$). At each stage i, a model is constructed by addition of $d_i^+ = d_i - d_{i-1}$ constraints to a model selected at stage $i-1$. For each model selected at stage $i-1$, we know from Remark 2 that bb_A constructs at most $b^{d_{i-1}^+}$ acceptable models. Since no more than m are selected at stage $i-1$, the total number of models constructed at stage i is $mb^{d_{i-1}^+}$. The total number of models constructed by the entire procedure is no more than $\sum_{i=1}^{k} mb^{d_i^+}$. That is, the total number of models constructed is $O(b^{d_{max}^+})$ where $d_{max}^+ = \max(d_1^+, d_2^+, \ldots, d_k^+)$.*

Models at a stage may not consist of a simple addition of constraints to those found earlier and we consider generalising models by splitting variables, before adding constraints (as shown in Fig. 37). For qualitative models, this translates to retaining the qualitative components found at the previous stage but disconnecting connections between some or all pairs whose outputs are connected together. While a general analysis will require a detailed description of the variable splitting procedure, a less detailed calculation is possible for the kinds of qualitative models sought here. The simplification results primarily from the "Distinct variables" restriction on well-posed models.

Remark 4 Incremental search space for qualitative models (limited generalisation). *Assume as before that the target model is restricted to d constraints and the maximum recall of any mode declaration is b. Assume further that model identification can be decomposed into $k \geq 1$ stages, with each stage resulting in models with $d_1, d_2, \ldots, d_k = d$ constraints (we will assume that all models in the initial set H_I have $d_0 \geq 0$ constraints and that $d_{i+1} \geq d_i$). We will now examine the effect of allowing generalisation by variable splitting only. For a model M selected at stage i, it is evident that if all variables in a constraint are distinct, then there can be at most $n_i = \max(d_i - 1, 0)$ co-references to any one variable v in M. Let the set of positions with co-references to a variable v be E_v. Variable splitting essentially renames variables at some or all of these positions into new ones. This is tantamount to partitioning the set E_v into equivalence classes, with positions in each equivalence class having the same variable; and each such partitioning giving rise to a model M' that is more general than m (in the sense that M' θ-subsumes Plotkin, 1970). The* nth *Bell number $B(n)$ gives the number of ways in which a set of size n can be partitioned into equivalence classes.*[7] *Thus, the number of models resulting from splitting variable co-references to a variable v in a model M from stage i is at most $B(n_i)$. If the maximum number of variables in any qualitative constraint is bounded by A, then there can be at most $s_i = Ad_i$ splittable variables in any model M from stage i. Therefore the number of models after generalisation of any M from stage i is at most $G(i) = B^{s_i}(n_i)$ Since there are no more than m models at any stage i, the total number of models after generalisation is no more than $mG(i)$. Recall each of these is then specialised by bb_A to construct a model for stage $i+1$. The total number of models constructed by the entire procedure is thus no more than $\sum_{i=1}^{k} mG(i-1)\, b^{d_i^+}$.*

7. The *n*th Bell number is equal to $\sum_{k=0}^{n} S_n^{(k)}$. $S_n^{(k)}$, or Stirling numbers of the second kind, describe the way a set of *n* elements can be partitioned into *k* disjoint, non-empty subsets. These can be computed using the formula $S_n^{(k)} = S_{n-1}^{(k-1)} + kS_{n-1}^{(k)}$ (with $S_n^{(1)} = 1$).

In general, it is evident that variable splitting is not the only form of generalisation that may be needed: components found at a previous stage may have to be discarded entirely before constructing a model for the current stage. It is evident that allowing this form of generalisation will significantly increase the worst-case search complexity.

Remark 5 Incremental search space for qualitative models (general case). *Assume as before that the target model is restricted to d constraints and the maximum recall of any mode declaration is b. Assume further that model identification can be decomposed into $k \geq 1$ stages, with each stage resulting in models with $d_1, d_2, \ldots, d_k$ constraints (we will assume that all models in the initial set H_I have $d_0 \geq 0$ constraints and that $d_{i+1} \geq d_i$). From Remark 4 above, we know that the number of models after generalisation by splitting variables at stage i is $mG(i)$, each with d_i constraints. Each of these models can be generalised further by dropping one or more constraints. This results in a total of $mG(i)2^{d_i}$ models, each of which is then specialised by bb_A to construct a model for stage $i+1$. In the worst case, all the constraints found in each of the models at stage i are removed by the generalisation step and the specialisation step at stage $i+1$ has to construct models with $i+1$ constraints in each case. The total number of models constructed by the entire procedure is thus no more than $\sum_{i=1}^{k} mG(i-1)2^{d_{i-1}}\, b^{d_i}$.*

A.3 Application

In all cases, the application tasks are of a re-constructive nature. That is, a known target model for each stage of the incremental process is used to generate examples for that stage. These, along with the background knowledge and a set of random examples for the stage are given to the learner. (the random examples are needed for the Bayesian calculation described in the previous section) We then check to see if the target model is amongst the results returned by the learner. All experiments were conducted on a laptop equipped with a 1.5 GHz Intel Pentium M Processor and 768 MB of main memory. Examples are restricted to a random sample of no more than 500 observations of system behaviour and no more than 500 random observations (these are needed for the Bayesian cost calculations). Incremental construction of models was accomplished using ALEPH version 5, with the YAP compiler (version 5.0.1).

We describe here the background knowledge and examples relevant to each of the application tasks presented in the paper. Common to all tasks are definitions of the QSIM constraints. The definitions we use are based on those in Bratko (2001) and are available on request from the first author. In the following sections we describe the encoding of the examples, the mode declarations and the values of the main parameters used for each application task. The principal parameters for system identification are these: (1) The number of constraints in the model (the "size" constraint described in Section 5.1 on well-posed models); (2) Upper bound on the number of occurrences of any kind of constraint (the "language" constraint described in Section 5.1); (3) Upper bound on the the number of nodes to be seached (*n* in the *incsearch* procedure); (4) Upper bound on the the number of models to be selected from a stage (*m* in the *incsearch* procedure); (5) Upper bound on the number of new variables in any model (constraint 10 described in Section 5.1); (6) Upper bound on the number of irrelevant variables in any model (constraint 11 in Section 5.1).

A.3.1 THE TANK MODELS

System variables for the coupled tanks system are `La`, `Lb`, `InflowA`, `Fab` and `OutflowB`. Examples for both the coupled tanks and single tank system are encoded using a $state/5$ predicate (the argu-

```
state(l:0/inc,l:0/std,f:0...inf/std,f:0/inc,f:0/std).
state(l:0/inc,l:0...inf/dec,f:0...inf/std,f:minf...0/inc,f:0...inf/dec).
state(l:0...inf/dec,l:0/inc,f:0...inf/std,f:0...inf/dec,f:0/inc).
state(l:0...inf/dec,l:0...inf/dec,f:0...inf/std,f:0...inf/dec,f:0...inf/dec).
state(l:0...inf/dec,l:0...inf/dec,f:0...inf/std,f:0...inf/inc,f:0...inf/dec).
```

Figure 38: Example observations from the coupled tank system.

```
ADD(+level,+level,-level)   SUB(+level,+level,-level)
ADD(+flow,+flow,-flow)      SUB(+flow,+flow,-flow)

MPLUS(+level,-level)        MPLUS(+level,-flow)
MPLUS(+flow,-flow)          MPLUS(+flow,-level)

MMINUS(+level,-level)       MMINUS(+level,-flow)
MMINUS(+flow,-flow)         MMINUS(+flow,-level)

MINUS(+level,+level)        MINUS(+flow,+flow)

DERIV(+level,-flow)
```

Figure 39: Mode declarations used for identifying the tank system.

ments refer to the system variables `La`–`OutflowB`, in the order just listed). Some of the observations are shown in Fig. 38 (the syntax used is in the Prolog language):
22 observations are generated in all using the correct model for the coupled tank system. For the first stage of learning—the single tank system—observations made for Tank A (that is, `La` and `InflowA`) are ignored. This is achieved using the following mode declaration for $state/5$ (here, the "_" denotes that the corresponding argument is to be ignored):

```
STATE(_,+level,_,+flow,+flow)
```

In contrast, the mode declaration for $state/5$ for the coupled tank system is as follows:

```
STATE(+level,+level,+flow,+flow,+flow)
```

Mode declarations for the QSIM constraints for both single and coupled tanks are shown in Fig. 39.

The values of the principal parameters for the two stages are shown in the tabulation below.

Parameter	Stage 1 (single tank)	Stage 2 (coupled tanks)
Size	3	7
Language	2	2
n	1000	1000
m	1000	1000
Newvars	3	3
Irrelev	0	0

```
state(p:0...inf/dec,n:0...inf/inc).
state(p:0...inf/std,n:0...inf/inc).
state(p:0...inf/inc,n:0...inf/dec).
```

Figure 40: Example observations from the predator-prey system.

```
ADD(+qval,+qval,-qval)      SUB(+qval,+qval,-qval)

MPLUS(+predator,-qval)      MPLUS(+prey,-qval)
MPLUS(+qval,-qval)          MPLUS(+predator,+prey)

MMINUS(+predator,-qval)     MMINUS(+prey,-qval)
MMINUS(+qval,-qval)         MMINUS(+predator,+prey)

MINUS(+predator,+qval)      MINUS(+prey,+qval)
MINUS(+predator,+prey)      MINUS(+qval,+qval)

DERIV(+predator,-qval)
DERIV(+prey,-qval)
```

Figure 41: Mode declarations used for identifying the predator-prey system.

A.3.2 THE PREDATOR-PREY MODELS

System variables for the predator-prey system are the predator population `P` and the prey population `N`. Examples for the system are encoded using a *state*/2 predicate (the arguments of which are `P` and `N`). Some of the observations are shown in Fig. 40.
5 observations of system behaviour are obtained using the target model. The mode declaration for the *state*/2 predicate is

```
STATE(+predator,+prey)
```

Mode declarations for the QSIM constraints are shown in Fig. 41.

The values of principal parameters are shown in the tabulation below.

Parameter	Value
Size	6
Language	2
n	1000
m	1000
Newvars	5
Irrelev	0

A.3.3 THE LUNG MODELS

System variables for identifying the human lung model are `Pa`, `Va`, `Pi`, `Vid` and `Pv`. Examples for the system are encoded using a *state*/5 predicate (the arguments of which are `Pa`–`Pv` in the order just listed). Some of the observations are shown in Fig. 42.
500 observations of system behaviour are obtained using the target model. The mode declaration for the *state*/5 predicate is:

```
state(p:0/std,v:0/inc,p:0...inf/inc,f:0/inc,p:0/inc).
state(p:0/std,v:0...inf/dec,p:0/std,f:0...inf/dec,p:0/std).
state(p:0/std,v:0...inf/dec,p:0/inc,f:0/inc,p:0/inc).
state(p:0/std,v:0...inf/dec,p:0/inc,f:0...inf/std,p:0/inc).
state(p:0/std,v:0...inf/dec,p:0/inc,f:0...inf/inc,p:0/std).
```

Figure 42: Example observations from the lung system.

```
ADD(+press,+press,-press)          SUB(+press,+press,-press)
ADD(+vol,+vol,-vol)                SUB(+vol,+vol,-vol)
ADD(+volrate,+volrate,-volrate)    SUB(+volrate,+volrate,-volrate)
ADD(+qval,+qval,-qval)             SUB(+qval,+qval,-qval)

MPLUS(+press,-press)               MPLUS(+press,-vol)
MPLUS(+press,-volrate)             MPLUS(+press,-qval)
MPLUS(+vol,-vol)                   MPLUS(+vol,-volrate)
MPLUS(+vol,-qval)                  MPLUS(+qval,-volrate)
MPLUS(+qval,-qval)

MMINUS(+press,-press)              MMINUS(+press,-vol)
MMINUS(+press,-volrate)            MMINUS(+press,-qval)
MMINUS(+vol,-vol)                  MMINUS(+vol,-volrate)
MMINUS(+vol,-qval)                 MMINUS(+qval,-volrate)
MMINUS(+qval,-qval)

MINUS(+press,+press)               MINUS(+vol,+vol)
MINUS(+volrate,+volrate)           MINUS(+qval,+qval)

MULT(+press,+press,-qval)          MULT(+press,+vol,-qval)
MULT(+press,+volrate,-qval)        MULT(+press,+qval,-qval)
MULT(+vol,+vol,-qval)              MULT(+vol,+volrate,-qval)
MULT(+vol,+qval,-qval)             MULT(+qval,+volrate,-qval)
MULT(+qval,+qval,-qval)

DERIV(+qval,-qval)
```

Figure 43: Mode declarations used to identify the lung system.

```
STATE(+press,+vol,+press,+volrate,+press)
```

Mode declarations for the QSIM constraints are shown in Fig. 43.

The values of principal parameters are shown in the tabulation below.

Parameter	Value
Size	7
Language	2
n	1000
m	1000
Newvars	6
Irrelev	0

```
state(l:0...inf/dec,l:0...inf/inc,f:0...inf/dec,l:0...inf/inc,l:0...inf/std,f:0/inc).
state(l:0...inf/dec,l:0...inf/inc,f:0...inf/dec,l:0...inf/inc,l:0...inf/inc,f:0...inf/dec).
state(l:0...inf/dec,l:0...inf/inc,f:0...inf/dec,l:0...inf/inc,l:0...inf/inc,f:0...inf/inc).
state(l:0...inf/std,l:0...inf/std,f:0/std,l:0...inf/std,l:0...inf/std,f:0/std).
state(l:0...inf/dec,l:0...inf/dec,f:0...inf/std,f:0...inf/inc,f:0...inf/dec).
```

Figure 44: Example observations from the glucose-insulin regulatory system.

```
ADD(+glevel,+glevel,-glevel)   SUB(+ilevel,+ilevel,+iflow)
ADD(+ilevel,+ilevel,-ilevel)   SUB(+glevel,+glevel,+gflow)

MPLUS(+glevel,-glevel)         MPLUS(+glevel,-ilevel)
MPLUS(+ilevel,-glevel)         MPLUS(+ilevel,-ilevel)

SPLUS(+glevel,-glevel)         SPLUS(+glevel,-ilevel)
SPLUS(+ilevel,-glevel)         SPLUS(+ilevel,-ilevel)

MMINUS(+glevel,-glevel)        MMINUS(+glevel,-ilevel)
MMINUS(+ilevel,-glevel)        MMINUS(+ilevel,-ilevel)

SMINUS(+glevel,-glevel)        SMINUS(+glevel,-ilevel)
SMINUS(+ilevel,-glevel)        SMINUS(+ilevel,-ilevel)

MINUS(+glevel,+glevel)         MINUS(+ilevel,+ilevel)

DERIV(+glevel,+gflow)          DERIV(+ilevel,+iflow)
```

Figure 45: Mode declarations used for identifying the glucose-insulin models.

A.3.4 THE GLUCOSE-INSULIN MODELS

System variables for the glucose-insulin models are `Gin`, `G`, `DG`, `Iin`, `I`, and `DI`. Of these `DG` and `DI` are dependent on the glucose and insulin variables and could have been inferred from them. Examples for both the insulin and glucose stages are encoded using a *state*/6 predicate (the arguments refer to the system variables `Gin`–`DI`, in the order just listed). Some of the observations are shown in Fig. 44 (the syntax used is in the Prolog language):
24 observations are generated in all using the correct model for glucose-insulin regulation. For the first stage of learning—the insulin stage—observations relevant to glucose (that is, `Gin`, `G` and `DG`) are ignored. This is achieved using the following mode declaration for *state*/6 (here, the "_" denotes that the corresponding argument is to be ignored):

```
STATE(_,_,_,+ilevel,+ilevel,+iflow)
```

In contrast, the mode declaration for *state*/6 for the glucose stage is as follows:

```
STATE(+glevel,+glevel,+gflow,+ilevel,+ilevel,+iflow)
```

Mode declarations for the QSIM constraints for both insulin and glucose stages are shown in Fig. 45. The values of the principal parameters for the two stages are shown in the tabulation below.

```
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0/std,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0...inf/dec,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0...inf/std,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0...inf/inc,g6p:0/std,glc:0...inf/dec]).
glycolysis([adp:0/std,atp:0...inf/std,f16bp:0/std,f6p:0/std,g6p:0/std,glc:0...inf/std],
        [adp:0...inf/inc,atp:0...inf/dec,f16bp:0...inf/inc,f6p:0/std,g6p:0...inf/dec,glc:0...inf/dec]).
```

Figure 46: Example observations from the priming stage of glycolysis.

Parameter	Stage 1 (insulin)	Stage 2 (glucose)
Size	3	9
Language	2	2
n	1000	1000
m	1000	1000
Newvars	5	5
Irrelev	0	0

A.3.5 THE GLYCOLYSIS MODELS

System variables for identifying models at any stage of glycolysis are cell-states before and after the stage. Examples for a stage are encoded using a *glycolysis*/2 predicate (the arguments of which are the cell-state before and after the reactions involved in that stage). Some of the observations for the first stage (priming) are shown in Fig. 46.
500 observations of system behaviour are obtained using the target models for each stage. The mode declaration for the *glycolysis*/2 predicate is:

```
GLYCOLYSIS(+cellstate,-cellstate)
```

Although QSIM constraints form the basis of qualitative reactions, the models constructed use a *pathway*/3 predicate. The mode declaration for this predicate is simply:

```
PATHWAY(+cellstate,#qreactions,-cellstate)
```

(Here the "#" indicates that a corresponding argument is a ground term: in this case a sequence of qualitative reactions).

The values of principal parameters for the three stages are shown in the tabulation below.

Parameter	Stage 1 (priming)	Stage 2 (splitting)	Stage 3 (phosphorylation)
Size	1	2	3
Language	3	3	3
n	1000	1000	1000
m	1000	1000	1000
Newvars	3	3	3
Irrelev	0	0	0

System Variables	Search Space
4	15
5	181
6	2163
7	27133
8	364395
9	5272861
10	82289163

Figure 47: Number of partition-sequences to be searched for a given number of system variables.

Appendix B. Automatic Decomposition

We present here a specification for the problem of automatic decomposition addressed in the latter half of the paper, along with implementation details of a search procedure that identifies an acceptable decomposition.

B.1 Specification

We will assume that we are looking to decompose a system specified by a set of qualitative system variables. The problem can be specified as follows. Given a non-empty set S of system variables, consider first the notion of a "partition-sequence" $(S_1, S_2, \ldots, S_n)$, in which $S_i \subset S$, and $S_1, S_2 \ldots S_n$ form a partition of S. Given a set E of observed values for the system variables S, background knowledge B, a refinement operator ρ, a cost function f, and a partition-sequence $P = (S_1, S_2, \ldots, S_n)$, we are able to construct a n-stage incremental learner of the form shown in Fig. 6(b) that returns a set of models $H_n = \mathsf{L}(B, E_n, \rho, f, H_{n-1})$ with minimal cost, where $H_0 = \{\emptyset\}$ and at each stage i, E_i are the values observed for variables $S_1 \cup S_2 \cdots S_i$. Let us assume that we are also able to obtain the cost of H_n, which, for reasons that will become obvious immediately, we call C_P. With automatic system decomposition, we are concerned with a procedure that returns an optimal partition-sequence P^* such that, of all possible partition sequences P, $C_{P^*} \leq C_P$ (that is, P^* yields models with the least cost).

Remark 6 Search space for decompositions. *We are able to provide some details on the combinatorics of the search for decompositions. Recall that the number of partitions of a set of n elements into exactly k non-empty blocks is given by $S_n^{(k)}$ (Stirling's number of the second kind). For each such partition, a valid answer is given by an ordering of the blocks into some sequence. The number of k-length partition-sequences is thus $D_n^{(k)} = k!S_n^{(k)}$. We have an additional constraint that requires the first block in any partition-sequence to contain at least 2 elements. This means that we are only interested in partition-sequences of length $1, 2, \ldots, n-2$. The total number of partition-sequences to be considered is therefore $D_n^{(1)} + D_n^{(2)} + \cdots + D_n^{(n-2)}$. This is at most $(n-2)D_n^{(n-2)}$. Of course, models have be constructed with each element of any partition-sequence. The complexity of this has be estimated in the previous section.*

Fig. 47 tabulates the size of the search space for some values of n (the number of system variables), showing how an addition of a system variable increases the size of the search space by an order of magnitude (the number appears always to be greater than 10^{n-3}).

B.2 Implementation

Figure 48 shows a GSAT-like randomised local search procedure that identifies a partition-sequence based on a greedy selection of elements. In experiments for the paper we have made one modification to this procedure, which we have not shown in the figure for simplicity. In Fig. 48 variable subsets are compared simply on costs. If a pair of variable subsets V_1 and V_2 have the same cost, we further examine the following. For each of $V_{1,2}$ we obtain a best-case estimate of the length of the final partition-sequence. The subset with the shorter length is preferred. If these lengths are also the same, then the subset with fewer variables is preferred (on the assumption that the resulting ILP model would be simpler). In addition, of course, we will use $\rho = \rho_A$ and $f = f_{Bayes}$.

Remark 7 Space searched by *rls***.** *Let $|S| = n$. The procedure contains 3 loops in Steps 4, 4f, and 4(f)viii The loop in Step 4(f)viii iterates at most M times. On each iteration, there are at most n local moves from any subset. Therefore, at most Mn subsets are examined by the loop in Step 4(f)viii. This is called no more than R times by the loop in Step 4f, resulting in at most MRn subsets. The outermost loop in Step 4 can iterate no more than n times, which means the number of subsets examined are at most MRn^2.*

B.3 Application

We consider first 3 artificial problems obtained by a random decomposition of a set of 10 variables into 2, 3 and 4 stages. For each problem and stage, the "correct" variable subset is assigned the least cost possible. All other subsets are assigned costs randomly. The task is to find the correct variable subset at each stage for each of the 3 problems.[8] The search space has approximately 10^8 elements (in contrast to the coupled tanks problem, which has about 10^2). Figures 49 summarises the results of attempting to identify the correct decompositions for each of the 3 problems.

More generally, we examine the values of R and M needed for identifying the correct decomposition.[9] The values required using artificial problem sets of the kind just described, with $n = 4, 5, 6, 8,$ and 10 variables with $k = 2, 3,$ and 4 stages are shown in Fig. 50. The probability of obtaining the correct decomposition are shown in Fig. 51.

From these results, it is evident that both an increase in the number of variables or the number of stages usually requires an increase in the values of R and M (Fig. 50a), and that as the values of R and M are increased, the probability of obtaining the correct decomposition increases (Fig. 50b). Both these observations may be evident to the reader since an increase in either the number of variables or stages makes the search space larger. Increasing R and M then allows a more extensive search. A further characteristic of the automatic decomposition problem that may not be as obvious is this: since the number of variables left at each stage less than at the previous stage, we should, in principle, be able to achieve the same performance by starting with high values of R and M and progressively reducing their values after each stage. We do not explore this further, as such a progressive reduction is not a feature of the procedure here.

8. The procedure in the previous section has to be modified slightly, since there is no need to construct models using an ILP learner for these problems.
9. We note that the experiments are concerned with exact identification of the correct decomposition. Approximate identification, not addressed here, would yield higher probabilities.

$rls(S, B, E, \rho, f, R, M)$: Given a non-empty set of system variables S; background knowledge B; a set of values for the system variables E; a refinement operator ρ; a cost function f; an upper bound on the number of restarts R; and an upper bound on the depth of local moves M, returns a partition-sequence $(S_1, S_2, \ldots, S_k)$, in which each S_i results in the lowest cost model at stage i, given models constructed for S_{i-1}.

1. $i = 0$
2. $H_i = \{\emptyset\}$, $S_i = \emptyset$
3. $VarsLeft = S$
4. while $VarsLeft \neq \emptyset$ do
   (a) Increment i
   (b) $VarsUsed = S_0 \cup S_1 \cdots S_{i-1}$
   (c) $bestcost = \infty$
   (d) $VarsSelected = VarsLeft$
   (e) $r = 0$
   (f) while $r < R$ do
      i. $VarsAvail = VarsLeft \setminus VarsUsed$
      ii. Randomly select $V \subset VarsAvail$
      iii. Let c be the cost of the models returned by an incremental learner constructed using H_{i-1}, B, E, $VarsUsed \cup V$, ρ, f
      iv. if $c < bestcost$ then
         A. $bestcost = c$
         B. $VarsSelected = V$
      v. endif
      vi. $bestlocal = V$
      vii. $m = 0$
      viii. while $m < M$ do
         A. Let L be the set of all variable subsets constituting local moves from $bestlocal$
         B. Let V' be the element of L resulting in the least cost c' using an incremental learner constructed using H_{i-1}, B, E, $VarsUsed \cup V'$, ρ, f
         C. $bestlocal = V'$
      ix. if $c' < bestcost$ then
         A. $bestcost = c'$
         B. $VarsSelected = V'$
      x. endif
      xi. Increment m
      xii. endwhile
   (g) Increment r
   (h) endwhile
5. $S_i = VarsSelected$
6. $VarsLeft = VarsLeft \setminus VarsSelected$
7. endwhile

return $(S_1, S_2, \ldots, S_i)$

Figure 48: A randomised local search procedure for decomposing a set of system variables into a partition-sequence from which an incremental ILP learner can be constructed.

R	M			
	3	10	25	50
3	0.00	0.15	0.12	0.10
10	0.30	0.48	0.64	0.64
25	0.30	0.80	0.90	0.90
50	0.50	0.80	1.00	1.00

(a) 2-stage decomposition $(\{4,6,8,9,10\},\{1,2,3,5,7\})$

R	M			
	3	10	25	50
3	0.03	0.18	0.18	0.10
10	0.20	0.30	0.60	0.60
25	0.60	0.60	0.80	0.90
50	0.80	1.00	0.90	1.00

(a) 3-stage decomposition $(\{1,5,6,8,10\},\{4,9\},\{2,3,7\})$

R	M			
	3	10	25	50
3	0.03	0.00	0.00	0.00
10	0.00	0.18	0.04	0.32
25	0.25	0.0.36	0.40	0.40
50	0.18	0.63	0.72	0.64

(a) 4-stage decomposition $(\{2,3,4\},\{10\},\{6,7,8,9\},\{1,5\})$

Figure 49: Probability estimates of identifying the correct decomposition using the randomised local search procedure on artificial problems with a set of 10 variables $S = \{1,2,\ldots,10\}$. The decompositions to be identified are shown as sequences $(S_1,S_2,\ldots,S_k)$ where $S_i \subset S$, $S_i \cap S_j = \emptyset$ and k represents the number of stages. The probability estimates required for each stage are obtained from 30 trials of using the randomised procedure.

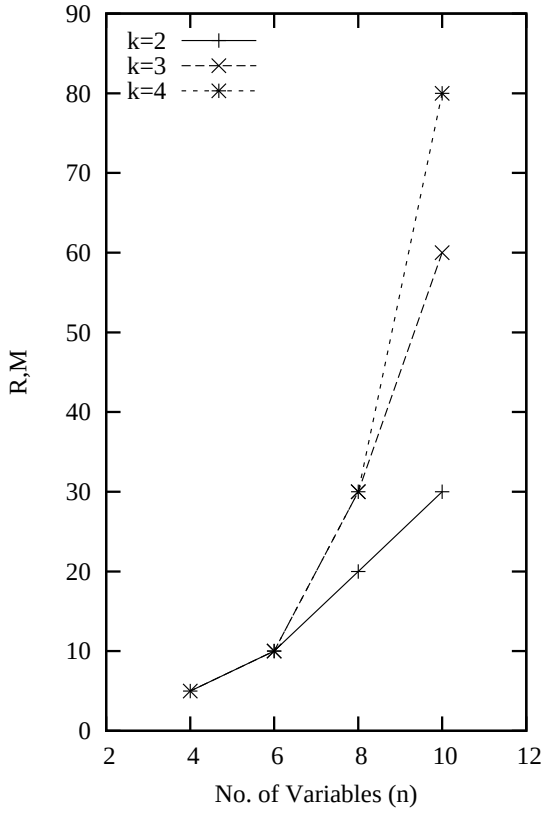


Figure 50: (a) R, M estimates for identification of the correct decomposition of a set of n variables into k stages.

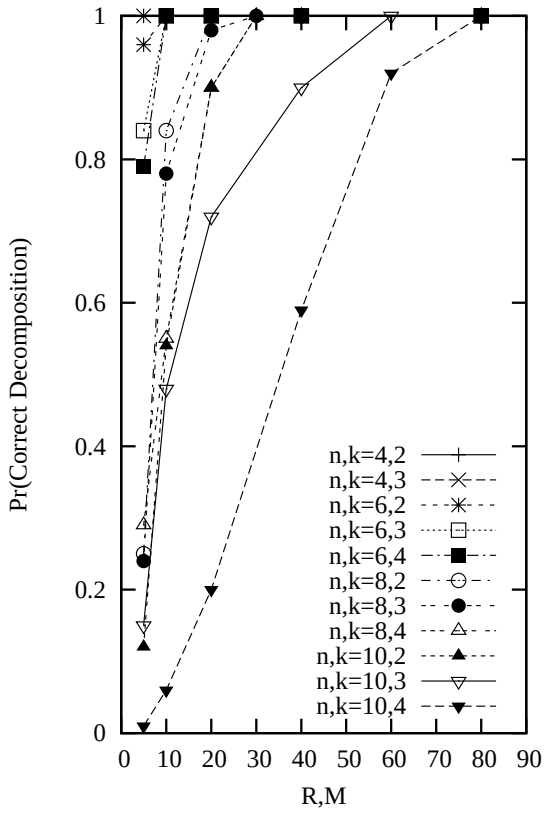


Figure 51: Probability of identifying the correct decomposition for specific values of n, k as a function of R, M.

References

R.L. Ashenhurst. The decomposition of switching functions. In *Proceedings of an International Symposium on the Theory of Switching*, pages 74–116. Harvard University, 1957. (The earliest report with this title by the author is in a Bell Telephone Labs Report No. BL-1(11), in 1952.).

E. Boros, V. Gurvich, P.F. Hammer, T. Ibaraki, and A. Kogan. Decomposition of partially defined Boolean functions. Technical Report 94-9, DIMACS, March 1994.

I. Bratko. *Prolog Programming for Artificial Intelligence*. Addison-Wesley (3rd edition), London, 2001.

I. Bratko, I. Mozetic, and N. Lavrac. *Kardio: A Study in Deep and Qualitative Knowledge for Expert Systems*. MIT Press, Cambridge, 1989.

I. Bratko, S. Muggleton, and A. Varsek. Learning qualitative models of dynamic systems. In S. Muggleton, editor, *Inductive Logic Programming*, pages 437–452. Academic Press, London, 1992.

D.J. Clancy and B. Kuipers. Model decomposition and simulation. In *Proceedings of the Eighth International Workshop on Qualitative Physics about Physical Systems (QR-94)*, Nara, Japan, 1994.

G.M. Coghill, S.M. Garrett, and R.D. King. Learning qualitative models in the presence of noise. In *Proceedings of the QR'02 Workshop on Qualitative Reasoning, Sitges, Spain*, 2002.

G.M. Coghill, S.M. Garrett, A. Srinivasan, and R.D. King. Qualitative system identification from imperfect data. Technical Report AUCS/TR0501, University of Aberdeen, Aberdeen, 2005.

E. W. Coiera. Generating qualitative models from example behaviours. Technical Report 8901, University of New South Wales, Deptartment of Computer Science, May 1989a.

E. W. Coiera. Learning qualitative models from example behaviours. In *Proc. Third Workshop on Qualitative Physics*, pages 45–51, Stanford, August 1989b.

P.-J. Courtois. On time and space decomposition of complex structures. *Communications of the ACM*, 28(6):590–603, June 1985.

H.A. Curtis. *A New Approach to the Design of Switching Circuits*. Van Nostrand, Princeton, NJ., 1962.

D. Hau and E. Coiera. Learning qualitative models of dynamic systems. *Machine Learning Journal*, 26:177–211, 1997. Special Issue on ILP.

A. L. Hodgkin and A. F. Huxley. A quatitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

L. Ironi and M. Stefanelli. In *Proceedings of the Eighth International Workshop on Qualitative Physics about Physical Systems (QR-94)*, Nara, Japan, 1994.

Y. Iwasaki and H. A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986. See also De Kleer and Brown's rebuttal and Iwasaki and Simon's reply to their rebuttal in the same volume of this journal.

R.D. King, S.M. Garrett, and G.M. Coghill. On the use of qualitative reasoning to simulate and identify metabolic pathways. *Bioinformatics*, 21:2017–2026, 2005.

B. Kuipers. *Qualitative Reasoning*. MIT Press, 1994.

W.P. Kuo, T-K. Jenssen, A.J. Butte, L. Ohne-Machado, and I.S. Kohane. Analysis of matched mRNA measurements from two different microarray technologies. *Bioinformatics*, 18(2):405–412, 2002.

Y. Lazebnik. Can a biologist fix a radio? Or, what I learned while studying apoptosis. *Cancer Cell*, 2:179–182, 2002.

E. McCreath. *Induction in First Order Logic from Noisy Training Examples and Fixed Example Sizes*. University of New South Wales (PhD. Thesis), Sydney, 1999.

I. Mozetic. Learning of qualitative models. In I. Bratko and Nada Lavrac, editors, *Progress in Machine Learning: Proceedings of EWSL '87: 2nd European Working Session on Learning*, pages 201–217. Sigma Press, 1987.

S. Muggleton. Inductive logic programming: derivations, successes and shortcomings. *SIGART Bulletin*, 5(1):5–11, 1994.

S. Muggleton. Inverse entailment and Progol. *New Gen. Comput.*, 13:245–286, 1995.

S. Muggleton. Learning from positive data. In *Proceedings of the Sixth Inductive Logic Programming Workshop*, LNAI, pages 358–376, Berlin, 1996. Springer-Verlag.

S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

J. D. Murray. *Mathematical Biology*. Springer, Berlin, 1993. Vol. 19, Biomathematics Texts Series, 2nd edition.

C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimisation*. Prentice-Hall, Edgewood-Cliffs, NJ, 1982.

D. Paulson and Y. Wand. An automated approach to information systems decomposition. *IEEE Transactions on Software Engineering*, 18:174–189, 1992.

G.D. Plotkin. A note on inductive generalisation. In B. Meltzer and D. Michie, editors, *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York, 1970.

B. L. Richards, I. Kraan, and B. J. Kuipers. Automatic abduction of qualitative models. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92)*, pages 723–728, San Jose, CA, July 1992.

F. Riguzzi. Two results regarding refinement operators. Technical Report TUM-I0510, Technische Universität Müenchen, Munich, 2005.

A. C. C. Say and S. Kuru. Qualitative system identification: deriving structure from behavior. *Artificial Intelligence*, 83:75–141, 1996.

B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*. AAAI Press, 1994.

B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 440–446. AAAI Press, 1992.

A.D. Shapiro. *Structured Induction in Expert Systems*. Addison-Wesley, Wokingham, 1987.

M. Shpak, P.F. Stadler, G.P. Wagner, and L. Altenberg. Simon-Ando decomposability and fitness landscapes. *Theory in Biosciences*, 2004a.

M. Shpak, P.F. Stadler, G.P. Wagner, and J. Hermisson. Aggregation of variables and system decomposition: applications to fitness landscapes. *Theory in Biosciences*, 2004b.

H. A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

T. Soderstrom and P. Stoica. *System Identification*. Prentice Hall, 1989.

A. Srinivasan. The Aleph manual. Available at http://www.comlab.ox.ac.uk/oucl/ research/areas/machlearn/Aleph/, 1999.

A. Srinivasan and R. Kothari. A study of applying dimensionality reduction to restrict the size of a hypothesis space. In *Proceedings of the Fifteenth International Conference on Inductive Logic Programming (ILP2005)*, LNAI 3625, pages 348–365, Berlin, 2005. Springer.

D. Suc, D. Vladusic, and I. Bratko. Qualitatively faithful quantitative prediction. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1052–1057. Morgan Kaufmann, 2003.

L. Todorovski and S. Džeroski. Using domain knowledge on population dynamics modelling for equation discovery. In *Proceedings of the Twelfth European Conference on Machine Learning*, pages 478–490. Springer (LNCS 2167), 2001.

L. Todorovski, S. Dzeroski, A. Srinivasan, J. Whiteley, and D. Gavaghan. Discovering the structure of partial differential equations from example behavior. In *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, 2000. Morgan Kaufmann. URL `ftp://ftp.comlab.ox.ac.uk/pub/Packages/ILP/Papers/AS/pde.ps.gz`.

A.M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B (London)*, 237:37–72, 1952.

W.Bialek and D.Botstein. *Science*, 2004.

M. P. Wellman. Qualitative simulation with multivariate constraints. In *Proc. of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 547–557, 1991.

Journal of Machine Learning Research 9 (2008) 1535-1558 Submitted 12/07; Revised 6/08; Published 7/08

Learning to Combine Motor Primitives Via Greedy Additive Regression

Manu Chhabra MCHHABRA@CS.ROCHESTER.EDU
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA

Robert A. Jacobs ROBBIE@BCS.ROCHESTER.EDU
Department of Brain & Cognitive Sciences
University of Rochester
Rochester, NY 14627, USA

Editor: Peter Dayan

Abstract

The computational complexities arising in motor control can be ameliorated through the use of a library of motor synergies. We present a new model, referred to as the Greedy Additive Regression (GAR) model, for learning a library of torque sequences, and for learning the coefficients of a linear combination of sequences minimizing a cost function. From the perspective of numerical optimization, the GAR model is interesting because it creates a library of "local features"—each sequence in the library is a solution to a single training task—and learns to combine these sequences using a local optimization procedure, namely, additive regression. We speculate that learners with local representational primitives and local optimization procedures will show good performance on nonlinear tasks. The GAR model is also interesting from the perspective of motor control because it outperforms several competing models. Results using a simulated two-joint arm suggest that the GAR model consistently shows excellent performance in the sense that it rapidly learns to perform novel, complex motor tasks. Moreover, its library is overcomplete and sparse, meaning that only a small fraction of the stored torque sequences are used when learning a new movement. The library is also robust in the sense that, after an initial training period, nearly all novel movements can be learned as additive combinations of sequences in the library, and in the sense that it shows good generalization when an arm's dynamics are altered between training and test conditions, such as when a payload is added to the arm. Lastly, the GAR model works well regardless of whether motor tasks are specified in joint space or Cartesian space. We conclude that learning techniques using local primitives and optimization procedures are viable and potentially important methods for motor control and possibly other domains, and that these techniques deserve further examination by the artificial intelligence and cognitive science communities.

Keywords: additive regression, motor primitives, sparse representations

1. Introduction

To appreciate why motor control is difficult, it is useful to quantify its computational complexity. Consider, for example, an agent whose goal is to apply torques to each joint of a two-joint arm so that the endpoint of the arm moves from an initial location to a target location in 100 time steps. Also suppose that torques are discretized to one of ten possible values. In this case, the agent needs

to choose one sequence of torques from a set of 10^{200} possible sequences. Searching this set of possible sequences is clearly a computationally intractable problem.

To ameliorate the computational challenges arising in motor control, it has been hypothesized that biological organisms use "motor synergies" (Bernstein, 1967). A motor synergy is a dependency among the dimensions or parameters of a motor system. For example, a coupling of the torques applied at the shoulder and elbow joints would be a motor synergy. Motor synergies are useful because they reduce the number of parameters that must be independently controlled, thereby making motor control significantly easier (Bernstein, 1967). Moreover, synergies are often regarded as "motor primitives". For our current purposes, we focus here on frameworks in which an agent with a library of motor synergies quickly learns to perform complex motor tasks by linearly combining its synergies. This idea has motivated a great deal of neuroscientific research. For example, Mussa-Ivaldi, Giszter, and Bizzi (1994) identified frogs' motor synergies by stimulating sites in the frogs' spinal cords. Importantly, these authors verified that stimulation of two sites leads to the vector summation of the forces generated by stimulating each site separately.

The idea of characterizing complex movements as linear combinations of motor synergies has also been influential in the fields of artificial intelligence and cognitive science. In these fields, an important research question is how to build an agent with a useful set of synergies or, alternatively, how an agent can learn a useful set of synergies. Approaches to these issues are typically based on techniques from the machine learning literature. Some researchers have developed theories motivated by kernel-based techniques. For example, Thoroughman and Shadmehr (2000) studied the errors in people's reaching movements and concluded that humans learn the dynamics of reaching movements by combining primitives that have Gaussian-like tuning functions. Other researchers have speculated that motor primitives can be learned using dimensionality-reduction techniques. For example, Sanger (1995) analyzed people's cursive handwriting using principal component analysis (PCA) to discover their motor synergies. He showed that linear combinations of these synergies closely reconstructed human handwriting. Other examples using dimensionality-reduction to learn motor primitives include Chhabra and Jacobs (2006), d'Avella, Saltiel, and Bizzi (2003), Fod, Matarić, and Jenkins (2002), Jenkins and Matarić (2004), Safanova, Hodgins, and Pollard (2004), Sanger (1994), and Todorov and Ghahramani (2003, 2004).[1]

The fact that novel motor tasks can often be performed by linearly combining motor synergies is a surprising result. To see why this result is unexpected, consider the case in which an agent needs to control a two-joint arm to perform a motor task. Suppose that a task is defined as a sequence of desired joint angles (i.e., desired angles for the shoulder and elbow joints at each time step of a movement), that a cost function is defined as the sum of squared error between the desired and actual joint angles, and that the agent has a library of motor synergies where a synergy is a sequence of torques (i.e., torques for the shoulder and elbow joints at each time step of a movement). The agent attempts to perform the motor task by finding a set of coefficients for a linear combination of synergies minimizing the cost function. (This optimization might be performed, for example, using a gradient descent procedure, known as policy gradient, in which an agent uses the gradient of the cost function with respect to the coefficients, Sutton, McAllester, Singh, and Mansour, 1999;

1. Our review focuses on frameworks in which complex movements are expressed as linear combinations of motor primitives. There are, of course, frameworks that use motor primitives in other ways. A reader interested in this topic may want to see Bentivegna (2004), Ijspeert, Nakanishi, and Schaal (2003), Lau and Kuffner (2005), Lee, Chai, Reitsma, Hodgins, and Pollard (2002), Peters and Schaal (2006), and Stolle and Atkeson (2006), among other articles.

Williams, 1992.) Should we expect that a good set of linear coefficients—a set that leads to a near-zero value of the cost function—will exist and, if so, be easy to find? We believe that the answer is no. Recall that synergies are defined in terms of torques, the agent **linearly** combines synergies, the cost function is defined in terms of joint angles, and there is a **nonlinear** relationship between torques and joint angles. Finding a good set of linear coefficients should be a difficult optimization problem because the nonlinear function relating coefficient values to values of the cost function will contain many local minima. As a matter of terminology, we refer to this as the "Motor Basis Optimization Problem".

The Motor Basis Optimization Problem motivates the need to think about good ways of constructing a library of synergies, and good ways of learning to linearly combine the synergies to perform novel motor tasks. In this paper, we propose a new learning model that learns a sparse and overcomplete representation of the space of potentially useful motor commands, and learns to linearly combine elements of this representation using a "greedy additive regression" procedure. At a high level of abstraction, our procedure closely resembles the use of greedy additive schemes for feature selection in recent machine learning systems (e.g., Perkins, Lacker, and Theiler, 2003; Viola and Jones, 2004). For example, Viola and Jones (2004) used AdaBoost (Freund and Schapire, 1997; Schapire, 1990) to create a fast and robust classifier for detecting faces in visual images. They started by creating a large library of image feature detectors. They then constructed a classifier in an additive manner. At each iteration, a new classifier was created by adding a feature detector to the old classifier. The feature detector that was added was the one whose use reduced the error of the old classifier by the largest amount. The end result after several iterations was a successful classifier with a sparse representation in the sense that it used relatively few feature detectors from the library.

This paper introduces a new learning model for motor control referred to as the Greedy Additive Regression (GAR) model. The GAR model maintains a library of torque sequences (i.e., motor synergies). If possible, the GAR model learns new movements by additively combining sequences in this library. If not possible, new movements are learned by other means (e.g., via feedback error learning). The torque sequences for these new movements are then added to the library. (Unlike Viola and Jones, we do not construct a library of primitives by hand. Instead, we learn the primitives in this library using the set of training tasks.)

We present results comparing the performances of the GAR model with those of another model, referred to as the PCA model, that can be regarded as a generic example from a large class of approaches commonly used in the artificial intelligence and cognitive science literatures. The PCA model learns a library of motor primitives using PCA, and finds coefficients for linearly combining the primitives using gradient descent. Whereas the PCA model often yields poor results, the GAR model consistently shows excellent performance. We find that the acquisition of new movements by the GAR model is rapid when the library is used. Moreover, the library is overcomplete and also sparse, meaning that only a small fraction of the stored torque sequences are used when learning a novel movement. The library is also robust in at least two different ways. First, after an initial training period, nearly all novel movements can be learned as additive combinations of sequences in the library. Consequently, learning from scratch via, for example, feedback error learning becomes rarer over time. Second, the library is also robust in the sense that it shows good generalization when an arm's dynamics are altered between training and testing conditions. If, for example, an arm is suddenly required to carry a payload during testing, torque sequences in the library can still be additively combined to rapidly learn new movements with this altered arm. We also demonstrate that the model works well regardless of whether motor tasks are specified in joint space or Cartesian

space. Based on these results, we believe that the GAR model contains several desirable properties, including a library which maintains a sparse and overcomplete representation of the space of potentially useful motor commands, and an additive regression optimization procedure which is fast and robust.

This article is organized as follows. Section 2 describes the two-joint arm that we simulated, and Section 3 describes the motor tasks that we used. Section 4 describes the Greedy Additive Regression model. Section 5 reports the simulation results comparing the performances of the GAR and PCA models under a variety of conditions. In Section 6, we briefly consider the performances of these models when the system to be controlled is a linear system with redundancy. Section 7 states our conclusions and directions for future research.

2. Simulated Two-Joint Arm

We simulated a two-joint arm that coarsely resembles a human arm (Li and Todorov, 2004). The arm can be written as a second-order nonlinear dynamical system (Hollerbach and Flash, 1982):

$$\mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta,\dot{\theta}) + \mathcal{B}\dot{\theta} = \tau$$

where τ is a vector of torques, $\theta, \dot{\theta}$, and $\ddot{\theta}$ are vectors of joint angle positions, velocities, and accelerations, respectively, $\mathcal{M}(\theta)$ is an inertial matrix, $\mathcal{C}(\theta,\dot{\theta})$ is a vector of Coriolis forces, and $\mathcal{B}$ is a joint friction matrix. The mathematical forms of these variables are as follows:

$$\begin{aligned}
\mathcal{M}(\theta) &= \begin{pmatrix} a_1 + 2a_2\cos\theta_2 & a_3 + a_2\cos\theta_2, \\ a_3 + a_2\cos\theta_2 & a_3 \end{pmatrix}, \\
\mathcal{C}(\theta,\dot{\theta}) &= \begin{pmatrix} -\dot{\theta_2}(2\dot{\theta_1} + \dot{\theta_2}), \\ \dot{\theta_1}^2 \end{pmatrix} a_2\sin\theta_2, \\
\mathcal{B}(\theta) &= \begin{pmatrix} b_{11} & b_{12}, \\ b_{21} & b_{22} \end{pmatrix}, \\
a_1 &= I_1 + I_2 + m_2 l_1^2, \\
a_2 &= m_2 l_1 s_2, \\
a_3 &= I_2
\end{aligned}$$

where I_1 and I_2 are the moments of inertia of the two links, m_1 and m_2 are the masses of the two links, and s_1 and s_2 are the distances from the joints to the links' center of masses. We used the same parameter values for the arm as Li and Todorov (2004). These values are given in Table 1.

3. Motor Tasks

A motor task is to apply torques to the arm so that it follows a desired trajectory defined in joint space. A desired trajectory is specified by a sequence of joint angles written as a 2×50 matrix of 2 joint angles over 50 time steps, where each time step corresponds to 7 ms of simulation. This trajectory is created in several stages (see Figure 1). First, we generate a trajectory in Cartesian space. To generate this trajectory, an initial position for the end-effector of the arm is chosen by randomly sampling each joint angle from a uniform distribution between 0 and $\pi/2$. Then a final position for the end-effector is chosen as the end point of a vector v of length d at an angle ψ starting

Constant	Value	Constant	Value
b_{11}	0.05 kgm^2s^{-1}	b_{22}	0.05 kgm^2s^{-1}
b_{21}	0.025 kgm^2s^{-1}	b_{12}	0.025 kgm^2s^{-1}
m_1	1.4 kg	m_2	1.0 kg
l_1	0.30 m	l_2	0.33 m
s_1	0.11 m	s_2	0.16 m
I_1	0.025 kgm^2	I_2	0.045 kgm^2

Table 1: Values of constants used in the simulation of a two-joint arm.

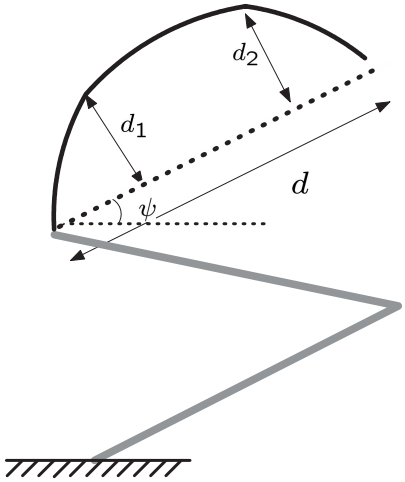


Figure 1: Schematic drawing depicting how a motor task is generated.

at the end-effector's initial position, where d and ψ are chosen uniformly at random between 10 cm and 30 cm, and between 0 and 2π, respectively. Next, two via points are chosen at distances d_1 and d_2 perpendicularly away from the vector v at locations $d/3$ and $2d/3$. Both d_1 and d_2 are drawn uniformly at random between -10 cm and 10 cm. Finally, a trajectory is generated by fitting a smooth cubic spline between the initial position, the two via points, and the final position. The Cartesian-space trajectory is converted to a joint-space trajectory by solving the robot arm's inverse-kinematics using the MATLAB robotics toolbox (Corke, 1996). The duration of movement is set to 350 ms, and the resulting joint-space trajectory is sampled at 7 ms intervals to get the 2×50 matrix defining a desired trajectory.

Given a desired joint-space trajectory, a motor task is to apply a time-varying torque to the arm so that the arm follows the desired trajectory. Torques are sampled every 7 ms, meaning that torques can be written as a 2×50 matrix. The cost function corresponding to the motor task is the sum of

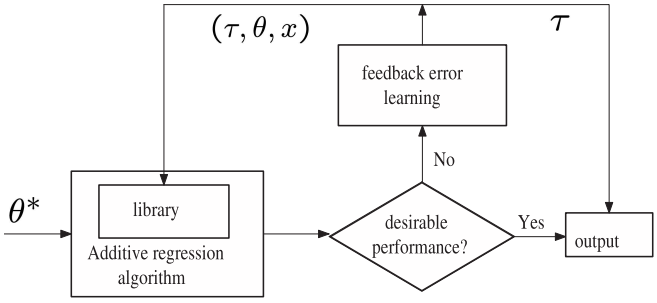


Figure 2: A schematic description of the Greedy Additive Regression(GAR) model. A desired trajectory θ^* is given as an input to the model. An additive regression algorithm is then used to construct a torque sequence by linearly combining sequences from the library. If this algorithm fails to find a linear combination yielding good performance, the model acquires a new torque sequence by other means (e.g., via feedback error learning), and then adds this new sequence to the library.

squared error between the desired and actual joint positions:

$$J = \sum_{t=1}^{50} \sum_{i=1}^{2} (\theta_i^*(t) - \theta_i(t))^2 \tag{1}$$

where $\theta_i^*(t)$ and $\theta_i(t)$ are the desired and actual angles of joint i at time t, respectively.

The motor tasks defined here are more complex than tasks often used in the literature in at least two respects. First, the desired Cartesian-space trajectories used here are typically highly curved, as opposed to straight-line reaching movements which are commonly used in experimental and computational studies of motor control. Second, our tasks specify desired joint angles at every time step. These tasks are more constrained than tasks that specify initial and final desired joint angles but allow an arm to have any joint angles at intermediate time steps.

4. The Greedy Additive Regression Model

We propose a model of motor learning called the Greedy Additive Regression (GAR) model. This model rapidly learns new motor tasks using a library of torque sequences. A schematic description of the model is given in Figure 2.

When a new motor task arrives, the model first checks whether a linear combination of sequences from the library achieves good performance on this task. Good performance is defined as a cost J less than ε (we set $\varepsilon = 0.05$ in our simulations). A potentially good linear combination is found via the additive regression algorithm which is described below. If a linear combination with good performance can be found, then this linear combination is used and nothing else needs to be done. If, however, such a linear combination is not found, then the model needs to learn a new torque sequence by other means. In the simulations reported in Section 5, we used feedback error learning to learn this new torque sequence (Kawato, Furukawa and Suzuki, 1987; see also Atkeson

and Reinkensmeyer, 1990, and Miller, Glanz, and Kraft, 1987).[2] The new torque sequence is then added to the library. Because a library has a fixed size of K, the addition of a new sequence may require the removal of an old sequence. Intuitively, the model removes the torque sequence that has been least used during the motor tasks that it has performed. Let n be an index over motor tasks, k be an index over sequences in the library, and $|\rho^k(n)|$ be the absolute value of the linear coefficient $\rho^k(n)$ assigned to sequence k on task n by the additive regression algorithm. The percent of the model's "total activation" that sequence j accounts for, denoted a^j, is defined as:

$$a^j = \frac{\sum_n |\rho^j(n)|}{\sum_n \sum_k |\rho^k(n)|} \times 100.$$

A sequence with large coefficients (based on magnitude, not sign) on many tasks would account for a large percent of the model's total activation, whereas a sequence with near-zero coefficients would account for a small percent. The model removes the torque sequence that accounts for the smallest percent of its total activation.

To complete the description of the GAR model, we need to describe the additive regression algorithm for finding potentially good linear combinations of torque sequences from the library for a motor task. As mentioned above, this algorithm is motivated by recent machine learning systems that have used greedy additive procedures for feature selection (Perkins, Lacker, and Theiler, 2003; Viola and Jones, 2004).

The additive regression algorithm is an iterative procedure. At iteration t, the algorithm maintains an aggregate torque sequence $F^{(t)}$ to perform a motor task such that:

$$F^{(t)} = \sum_{j=1}^{t} \rho_j f_j \tag{2}$$

where f_j is a sequence in the library and ρ_j is its corresponding coefficient. Note that the aggregate sequence $F^{(t)}$ is a weighted sum of t sequences from the library, but these sequences are not necessarily distinct. It is possible that the same sequence appears more than once in the summation in Equation 2. At each iteration of the algorithm, a sequence from the library is selected (with replacement), and a weighted version of this sequence is added to $F^{(t)}$ in order to create $F^{(t+1)}$. That is,

$$F^{(t+1)} = F^{(t)} + \rho_{t+1} f_{t+1} \tag{3}$$

where f_{t+1} is the library sequence selected to be added and ρ_{t+1} is its corresponding coefficient.

How does the algorithm choose f_{t+1} and ρ_{t+1}? Each torque sequence in the library is associated with a trajectory of joint angles. For computational convenience, the algorithm sets this trajectory to

2. In brief, feedback error learning proceeds as follows. An adaptive feedforward controller is used in conjunction with a fixed feedback controller. At each moment in time, the feedforward controller receives the desired joint positions, velocities, and accelerations, and produces a feedforward torque vector. The feedback controller receives the current and desired joint positions and velocities and produces a feedback torque vector. The sum of the feedforward and feedback torque vectors is applied to the arm, and the resulting joint accelerations are observed. During the learning portion of the time step, the inputs to the feedforward controller are set to the current joint positions, velocities, and accelerations, and the target output is set to the torque vector that was applied to the arm. This controller's parameters are then adapted so that it better approximates the mapping from the inputs to the target output in the future. Early in training, the outputs of the feedforward controller are near zero and most of the torques are supplied by the feedback controller. As training progresses, the feedforward controller better approximates the arm's inverse dynamics, and it supplies most of the torques. Feedback error learning is an attractive learning procedure because it is unsupervised; it does not require an external teacher but only a simple feedback controller.

a "prototypical" trajectory in the following sense. The position of the arm is initialized so that each joint angle is at its average initial value (i.e., each joint angle is initialized to $\pi/4$). The joint-angle trajectory associated with a torque sequence is then found by applying the sequence to the arm. A sequence is evaluated by correlating its joint-angle trajectory with $\partial J/\partial F^{(t)}$, the gradient of the cost function J with respect to the current aggregate torque sequence. This gradient indicates how the aggregate sequence should be modified so as to reduce the cost. In our simulations, it was obtained by numerically computing the partial derivative of the cost function with respect to each element of the aggregate sequence $F^{(t)}$.[3] The torque sequence whose trajectory is maximally correlated with this gradient, denoted f_{t+1}, is selected. To find the best coefficient ρ_{t+1} corresponding to this sequence, the algorithm performs a line search, meaning that the algorithm searches for the value of ρ_{t+1} that minimizes the cost $J(F^{(t)} + \rho_{t+1} f_{t+1})$ (we implemented a golden section line search; see Press, Teukolsky, Vetterling, and Flannery, 1992, for details). $F^{(t+1)}$ is then generated according to Equation 3, and the optimization proceeds to the next iteration. This process is continued until the value of the cost function converges (see Algorithm 1).

There are several possible perspectives on the additive regression algorithm. The idea of greedily selecting the next primitive from a library has also been explored in the feature selection literature. For example, Perkins, Lacker, and Theiler (2003) used a gradient-based heuristic at each iteration of their learning procedure to select the best feature from a set of features to add to a classifier. Our work differs from their work in many details because the domain of motor control forces us to confront the complexities inherent in learning to control a dynamical system (see also Tassa, Erez, and Smart, 2008). In addition, an appealing aspect of our work is that we use the solutions from prior tasks to create a library of primitives. We find that this practice leads to an overcomplete representation of the control space. Overcomplete representations have been shown to be useful in a wide range of applications (e.g., Lewicki and Sejnowski, 2000; Smith and Lewicki, 2006). In addition, the additive regression algorithm can be seen as performing gradient descent where the direction of the gradient at each iteration is projected onto the library sequence whose trajectory is maximally correlated with this gradient. The algorithm then minimizes the cost function by optimizing the coefficient corresponding to this sequence. The algorithm can also be seen as performing a type of "functional gradient descent" via boosting (readers interested in this perspective should see Bühlmann, 2003, or Friedman, 2001). Lastly, the algorithm can be seen as using "matching pursuit" to identify the next library sequence to add to the aggregate sequence at each iteration (see Mallat and Zhang, 1993, for details).

5. Simulation Results

This section reports a number of results using the GAR model. We compare the performances of the GAR model with those of another model, referred to as the PCA model, that can be regarded as a generic example from a large class of approaches commonly used in the artificial intelligence and cognitive science literatures. The PCA model performs dimensionality-reduction via PCA to learn a library of motor primitives. When given a novel motor task, the PCA model learns to perform the

3. $F^{(t)}$ is a 2×50 matrix. The partial derivative of the cost function with respect to element (j,k) of $F^{(t)}$ was computed by evaluating the cost of $F_+^{(t)}$ and $F_-^{(t)}$, where $F_+^{(t)}$ is the same as $F^{(t)}$ except that its $(j,k)^{\text{th}}$ element is set to $F^{(t)}(j,k) + \delta$ (similarly, $F_-^{(t)}$ is set to $F^{(t)}(j,k) - \delta$; we set $\delta = 0.01$.) The partial derivative was then approximated by $\frac{J(F_+^{(t)}) - J(F_-^{(t)})}{2\delta}$.

input : A desired trajectory θ^*
assume : A library $\mathcal{L} = \{(f^k, \theta^k)\}$ of torque sequences and their corresponding trajectories
output : An aggregate torque sequence F that minimizes cost J
$t \leftarrow 0$; $F \leftarrow 0$;
repeat
  $t \leftarrow t+1$
  numerically compute $\bigtriangledown J = \frac{\partial J}{\partial F}$
  From the library $\mathcal{L}$, pick a sequence f^k such that $\bigtriangledown J$ and θ^k are maximally correlated
  $f_{t+1} \leftarrow f^k$
  do a line search to find ρ_{t+1} that minimizes $J(F + \rho_{t+1} f_{t+1})$
  $F \leftarrow F + \rho_{t+1} f_{t+1}$
until J converges
output F

Algorithm 1: Additive regression algorithm for finding a linear combination of torque sequences from the library.

task using a policy gradient optimization procedure (Sutton, McAllester, Singh, and Mansour, 1999; Williams 1992) to learn a set of coefficients for linearly combining the motor primitives. (We regard the PCA model as generic because we regard PCA and gradient descent as generic dimensionality-reduction and optimization procedures, respectively.)

5.1 GAR versus PCA

In the PCA model, the library of motor synergies was created as follows. We first generated 3000 motor tasks as described in Section 3, and then used feedback error learning to learn a torque sequence for each task. This gave us 3000 sequences, each defined by a matrix of size 2×50. We re-stacked the rows of each matrix to form a vector of size 1×100. This gave us 3000 vectors (or data points) lying in a 100-dimensional space. We then performed dimensionality reduction via PCA. The 100 principal components accounted for all the variance in the data and, thus, these components were used as the library for the PCA model. We refer to these components as PCA sequences.

To learn to perform a novel motor task from a test set, the PCA model searched for good linear combinations of the PCA sequences. This search was conducted using a policy gradient procedure (Sutton, McAllester, Singh, and Mansour, 1999; Williams 1992). The linear coefficients were initialized to random values. At each iteration of the procedure, the gradient of the cost function with respect to the coefficients was numerically computed, and a line search in the direction of the gradient was performed (a golden section search method was implemented; see Press, Teukolsky, Vetterling, and Flannery, 1992, for details). This process was repeated until the cost function converged.

The GAR model was implemented as follows. Its library of torque sequences was created by running the model on 3000 motor tasks. The model's library size was set to 100. The sequences in this library at the end of training are referred to as GAR sequences. To learn to perform a novel motor task from a test set, the GAR model learned to linearly combine the GAR sequences using the additive regression algorithm described above.

The PCA and GAR models are two possible combinations of ways of creating libraries—one can create libraries of either PCA or GAR sequences—and ways of linearly combining sequences from

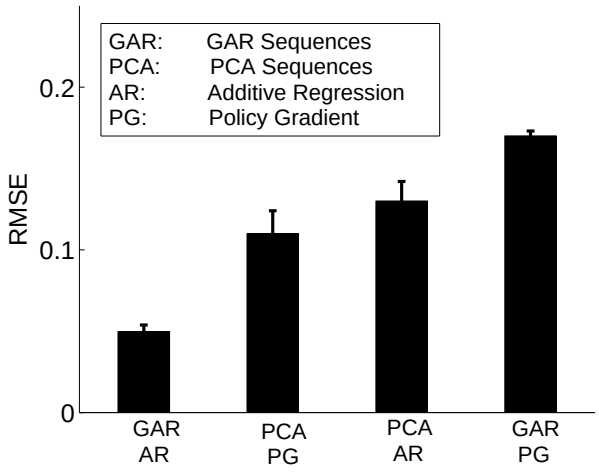


Figure 3: Average root mean squared errors of four systems on a test set of 100 novel motor tasks (the error bars show the standard errors of the means). The four systems use: (i) GAR sequences with additive regression (GAR model); (ii) PCA sequences with policy gradient (PCA model); (iii) PCA sequences with additive regression; and (iv) GAR sequences with policy gradient.

a library—one can learn linear coefficients through policy gradient or additive regression. The PCA model combines PCA sequences with policy gradient, whereas the GAR model combines GAR sequences with additive regression. For the sake of completeness, we also studied the remaining two combinations, namely, the combination of PCA sequences with additive regression and the combination of GAR sequences with policy gradient.

The results are shown in Figure 3. The horizontal axis gives a system's combination of library sequences and optimization technique. The vertical axis gives a system's average root mean squared error (RMSE where the error is between the desired and actual joint angles) on a test set of 100 novel motor tasks. Clearly, the GAR model (leftmost bar in figure) performed better than the PCA model (second bar from left). To further illustrate this point, the solid line in Figure 4 shows the Cartesian-space desired trajectory for a sample test task. The dashed line shows the trajectory achieved by the GAR model, and the dotted line shows the trajectory achieved by the PCA model. Whereas the GAR model found a curved trajectory that closely approximated the desired trajectory, the PCA model converged to a relatively straight-line movement which coarsely approximated the desired trajectory. Our simulation results suggest that this is a common outcome for the PCA model. It appears that the PCA model (and perhaps any system that uses policy gradient; see Figure 3) is prone to finding poor local minima of the error surface.

In addition to showing that the GAR model outperformed the PCA model, Figure 3 also shows that the GAR model outperformed the other systems considered here. Overall, the results are interesting because they suggest that it is not enough to choose a good library—consider that the system using GAR sequences with policy gradient performed poorly—and that it is also not enough to use a good optimization procedure—the system using PCA sequences with additive regression

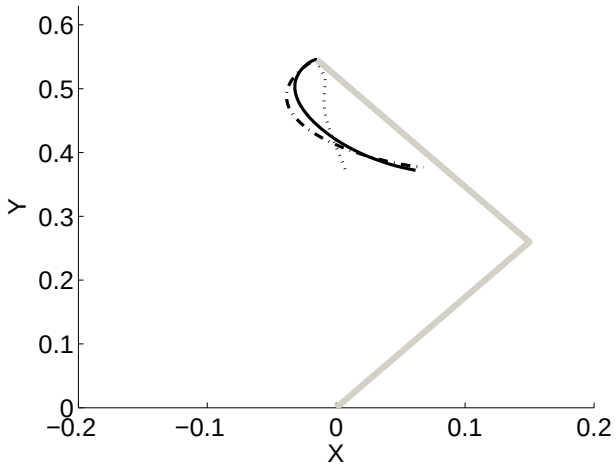


Figure 4: The solid line shows the Cartesian-space desired trajectory for a sample test task. The dashed line shows the trajectory achieved by the GAR model, and the dotted line shows the trajectory achieved by the PCA model.

performed poorly too. Instead, to achieve good performance it is necessary to consider the representational primitives and the optimization procedure as a pair. Representational primitives and optimization procedures are effective if a given procedure is able to find good solutions when the search space is based on these primitives.

Why does the GAR model work so well? Our results suggest that its due to its combination of "local" representational primitives (the GAR sequences) and a "local" optimization procedure (additive regression). To appreciate the coupling between representational primitives and optimization procedures, its important to keep in mind the differences between GAR and PCA sequences, and the differences between additive regression and gradient descent optimization procedures. Each individual GAR sequence is a solution to some task in the training set, whereas an individual PCA sequence is not necessarily a solution to a task but, rather, reflects properties of many tasks. In this sense, a GAR sequence can be regarded as a "local feature," and a PCA sequence can be regarded as a "global feature." Similarly, additive regression can be considered as a local optimization procedure because it adds at most one new feature to its linear combination at each iteration and because, at convergence, its linear combination tends to contain relatively few features. In contrast, gradient descent is a global optimization procedure because it finds linear combinations of all possible features. Because some features can have opposite effects, global optimization procedures lead to interference. Interference can be avoided by using a local optimization method. Local optimization methods have been shown to be effective in motor control in previous research. For example, Atkeson, Moore, and Schaal (1997) stored all previous experiences on control tasks in memory, and used a relatively local regression scheme (where locality was specified in terms of both space and time) to compute control signals for new tasks. They showed that their local learning method performed well, and also ameliorated the problem of global interference from features with opposing effects.

For linear systems and quadratic cost functions, we predict that the use of GAR versus PCA sequences, or additive regression versus gradient descent optimization procedures, should not matter much. Indeed, simulations on a linear system in Section 6 show that all four library/algorithm combinations work equally well. This is because a linear combination of either GAR or PCA sequences is, by linearity, a solution to some task. When searching for a good linear combination, a learner is searching among a set of task solutions for the particular solution which yields good performance on the current target task. This remains true regardless of whether a learner uses GAR or PCA sequences, or additive regression or gradient descent optimization procedures.

For nonlinear systems, however, this is not necessarily the case. With nonlinear systems, our results show that a learner using local primitives (which are task solutions) and local optimization procedures is preferable. This is because, when searching for a good linear combination, the local optimization procedure searches a set of combinations which are relatively close to solutions for some task. In the context of the GAR model, for example, we conjecture that each iteration of the additive regression procedure finds a linear combination of GAR sequences (again, each sequence is a solution to a task in the training set) which is itself close to a solution for some task due to the local nature of its search. In contrast, a global optimization procedure, such as gradient descent, would search among linear combinations which are far from any task solution. Finally, our results are consistent with empirical findings in the machine learning literature showing that additive schemes outperform gradient descent when searching for good linear combinations of features for novel classification tasks (Friedman, 2001; Perkins, Lacker, and Theiler, 2003; Viola and Jones, 2004).[4]

5.2 Visualizing Torque Sequences

The library of a GAR model is created on the basis of a wide variety of motor tasks. The torque sequences in the library should, therefore, be "representative" of the tasks they encode. Our goal here is to examine these sequences.

We trained a GAR model with 3000 training tasks using a library of size 100. We then ordered the sequences in the library by the percent of the model's total activation that a sequence accounted for. Figure 5 shows the Cartesian-space trajectories generated by the top three sequences. To generate these trajectories, the shoulder and elbow joint angles of the arm were initialized to $\pi/4$ and $\pi/2$ respectively. Each torque sequence was then applied to the arm, first with a coefficient of 1 and then with a coefficient of -1. Note that the trajectories span a wide range of directions. Several of the trajectories are highly curved, whereas others are closer to straight lines. This range is a result of the diverse set of tasks used to create the sequences. This graph illustrates that, even though the sequences are added to the library in an arbitrary order, the important sequences that remain in the library are representative of the motor tasks.

5.3 The GAR Model with Libraries of Different Sizes

Above we set the size of the library used by the GAR model to 100. Here we compare the model's performances with libraries of different sizes. If the size, denoted K, is too small, then torque sequences that are often useful for learning novel motor tasks might be removed. In contrast, if K is too big, then the library will contain many sequences which are nearly never used. Consequently, there ought to be an optimal value for K. We implemented the GAR model as described above

4. We thank an anonymous reviewer whose suggestions inspired these comments.

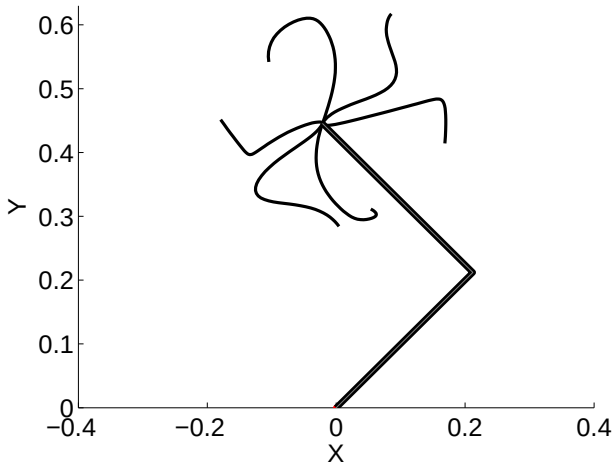


Figure 5: Cartesian-space trajectories generated by the three torque sequences that accounted for the largest percent of a GAR model's total activation. These trajectories were generated by initializing the shoulder and elbow joint angles of the arm to $\pi/4$ and $\pi/2$ respectively, and then applying the sequences to the arm with coefficients of 1 and -1.

using 3000 motor tasks. Three versions of the GAR model were used where the versions differed in the sizes of their libraries.

The results are shown in Figures 6 and 7. In Figure 6, the horizontal axis shows the number of motor tasks, and the vertical axis shows the percent of tasks in which a version of the GAR model needed to learn a torque sequence via feedback error learning. The latter value was obtained as follows. The motor tasks were divided into 60 blocks of 50 trials each. The percents for the blocks were then smoothed using a moving window of width 5. Results are reported for versions of the GAR model with library sizes of 50, 100, and 200. Early in training, a library has relatively few sequences, and feedback error learning must often be used. As training progresses, the library has many more useful sequences, and most novel motor tasks can be performed by linearly combining sequences from the library. In this case, feedback error learning is infrequently used. A comparison of the versions with different library sizes shows that the version with a library size of 50 used feedback error learning more often than versions with library sizes of 100 or 200. This suggests that a library size of 50 is too small.

From top to bottom, the three graphs in Figure 7 correspond to versions of the GAR model with library sizes of 50, 100, and 200. The sequences in a library are ordered according to the percent of a model's total activation that a sequence accounted for. The horizontal axis of each graph in Figure 7 plots the sequence number, and the vertical axis plots the percent of total activation that a sequence accounted for. The versions with libraries of size 100 and 200 show similar patterns of activation. In both cases, approximately the top 50 sequences accounted for nearly all the activation. The remaining sequences were rarely used. In contrast, the version with a library of size 50 had a different pattern of activation. Roughly all of the sequences in this library contributed to the model's total activation. We measured the average task error for each model (based on Equation 1) using

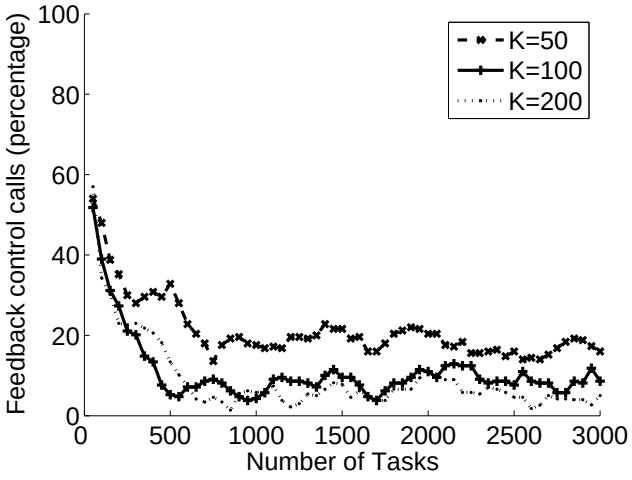


Figure 6: The horizontal axis shows the number of motor tasks, and the vertical axis shows the percent of tasks in which a version of the GAR model needed to learn a torque sequence via feedback error learning. The three curves in the figure correspond to versions of the GAR model with library sizes of 50, 100, and 200.

the last 1000 motor tasks. When $K = 50$, the average task error was 0.0925; when $K = 100$, the error was 0.0723; and when $K = 200$, the error was 0.0744. The corresponding standard error of the means were 0.010, 0.012, and 0.009. It seems that $K = 100$ is most efficient in the sense that it yielded good performance with a memory of moderate size. Furthermore, the version with $K = 100$ has the property that its use of sequences was relatively sparse. The top 10, 20, and 30 sequences accounted for 60, 78, and 88 percent of the version's total activation, respectively. Clearly, only a small fraction of the stored sequences tended to be used when learning a novel task.

5.4 GAR versus PCA in the Presence of Altered Dynamics

People are robust to changes in their arms' dynamics. For example, people can make accurate and smooth arm movements regardless of whether they carry no payload, a light payload, or a heavy payload. In this subsection, we compare the performances of the GAR and PCA models when they were trained without a payload, but a payload was added to the simulated arm during test trials.

The libraries for the GAR model (with a library of size 100) and the PCA model were created as described above with an arm that did not carry a payload. These models were then tested when the arm did carry a payload. Test trials were conducted as described above; that is, for each test task, a linear combination of torque sequences in a library was found via the additive regression algorithm for the GAR model, and via policy gradient for the PCA model. A set of 100 novel test tasks was generated. Models were evaluated on this set four times, once for each possible payload (payloads of 0, 1, 3, and 5 kg were used). Payloads were added to an arm by increasing the mass of the arm's elbow-wrist link (m_2 in Table 1). For the sake of completeness, we also tried the other two

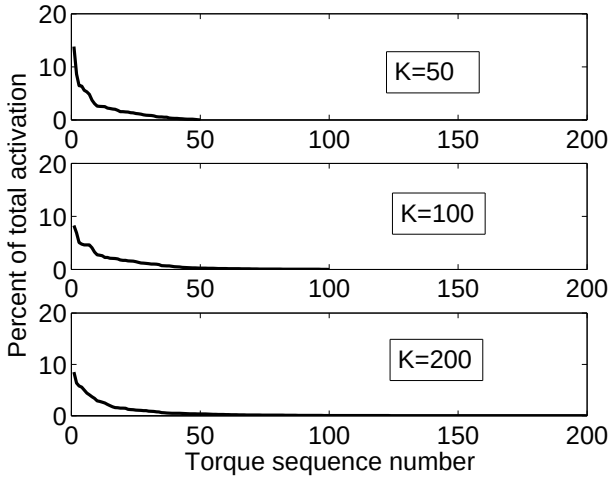


Figure 7: From top to bottom, the three graphs correspond to versions of the GAR model with library sizes of 50, 100, and 200. The sequences in a library are ordered according to the percent of a model's total activation that a sequence accounted for. The horizontal axis of each graph plots the sequence number, and the vertical axis plots the percent of total activation.

combinations of libraries with optimization algorithms, namely, the GAR sequences with policy gradient, and the PCA sequences with additive regression.

The results are shown in Figure 8. The vertical axis plots the average RMSE for each model for each payload. (The results for a payload of 0 are identical to those in Figure 3). For each payload, there are four bars corresponding to the four library/algorithm combinations. The performances of the PCA model degraded rapidly as the payload increased (2^{nd} bar in each set of bars). In contrast, the performances of the GAR model were robust (1^{st} bar in each set). We regard this successful generalization as a highly surprising result. It clearly demonstrates that the GAR model develops a useful library of torque sequences, and that the additive regression algorithm is a powerful optimization procedure for finding good linear combinations, even under test conditions that are very different from training conditions.

Why did the GAR model generalize so successfully? To address this question, we performed an additional analysis. The idea behind this analysis is to evaluate whether the GAR model generates similar libraries for different payloads. If this is the case, then additive regression should work well for tasks with novel payloads, even when using a library of GAR sequences constructed from zero-payload trials. We first generated a library of 100 GAR sequences using a training set of 3000 tasks where the simulated arm did not contain a payload. We then generated libraries for each non-zero payload using the same set of tasks. We compared each non-zero payload library to the zero-payload library. For each GAR sequence in a non-zero payload library, we found the sequence in the zero-payload library that was maximally correlated with this GAR sequence. For each non-zero payload, the average value of this maximum correlation is reported in Table 2. The GAR model successfully generalized from zero payloads to non-zero payloads because these correlations are large. The

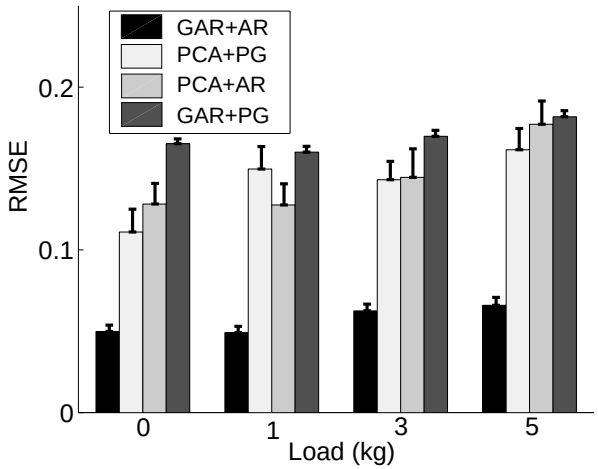


Figure 8: Average RMSEs of the GAR and PCA models on the test tasks when the arm carried different payloads.

Payload (kg)	Average maximum correlation	Standard error
1	0.84	0.08
3	0.81	0.04
5	0.73	0.06

Table 2: Average maximum correlation of the zero-payload library with the libraries built using non-zero payloads (see text for details).

other systems we evaluated were not able to take advantage of the similarities between solutions for zero-payload and non-zero payload tasks.

5.5 Motor Tasks Specified in Cartesian Space

In this subsection, we consider learning sequences for motor tasks when the desired trajectories are specified in Cartesian space instead of joint space. Using Cartesian trajectories adds an additional level of complexity. In addition to modeling the arm's inverse dynamics (a mapping from desired joint coordinates to torques), a system also needs to model the arm's inverse kinematics (a mapping from desired Cartesian coordinates to joint coordinates). An appealing feature of Cartesian trajectories is that they can be easily planned based on visuospatial information.

The cost function for this simulation is the sum of squared error between desired and actual positions of the arm's end-effector in Cartesian space:

$$J = \sum_{t=1}^{50} (r_x^*(t) - r_x(t))^2 + (r_y^*(t) - r_y(t))^2$$

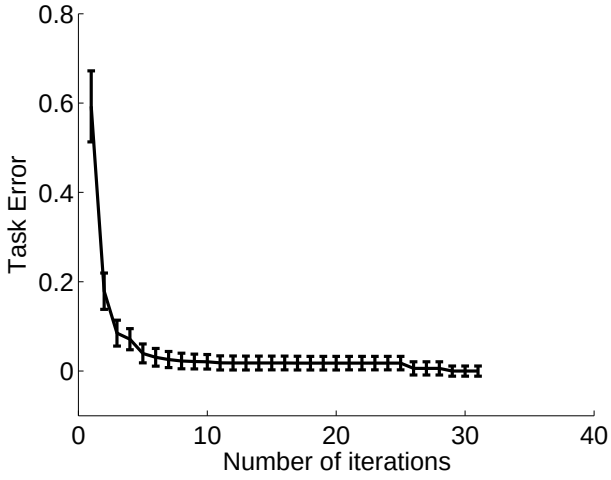


Figure 9: Results when motor tasks were specified in Cartesian space. The horizontal axis plots the number of iterations used by the GAR model, and the vertical axis show the average task error at each iteration.

where $(r_x^*(t), r_y^*(t))$ is the desired (x,y)-coordinates of the arm's end-effector in Cartesian space at time t, and $(r_x(t), r_y(t))$ is the actual coordinates. The GAR model was trained using 3000 motor tasks with a library of size 100. The library was constructed in the same way as before. An error threshold of $\varepsilon = 0.02$ was used to determine if a linear combination of torque sequences from the library provided a "good" aggregate sequence for a task (note that this is different from a threshold of $\varepsilon = 0.05$ used in previous simulations because the cost function is in different units now). We then created a test set of 100 tasks, and used the additive regression algorithm to learn a set of linear coefficients for each test task.

The results are shown in Figure 9. The horizontal axis plots the number of iterations used by the additive regression algorithm, and the vertical axis shows the average task error (Equation 8) at each iteration. Note that this error declined rapidly to a near-zero value. This outcome indicates that the GAR model has wide applicability in the sense that it is effective regardless of whether motor tasks are specified in joint space or Cartesian space.

6. GAR Model Applied to a Redundant and Unstable System

Until now, our simulations used a robotic arm. This section reports simulation results with a spring-mass system. In contrast to the robotic arm, this system allows us to evaluate different learners when the system to be controlled has linear dynamics, redundancy (three control signals move the system in a two-dimensional space), and is inherently unstable (zero or random control signals lead to divergent behavior). The system is schematically illustrated in Figure 10.

The spring-mass system has three elastic spring-like sticks that produce an opposing force when stretched or compressed. Stick 1 has resting length of l and connects the ground to point mass m_1. Stick 2 also has a resting length of l and connects m_1 to point mass m_2. Stick 3 has a resting length

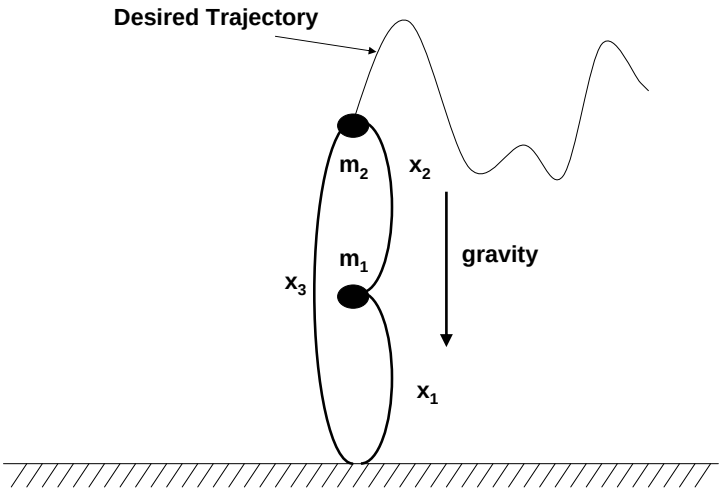


Figure 10: Schematic description of the spring-mass system used for the simulations.

of $2l$ and connects the ground to point mass m_2. All sticks have second-order linear dynamics. The dynamics of this system are governed by the following set of equations:

$$\begin{aligned} f_1 &= (l-x_1)k_1 + b_1\dot{x_1} + h_1u_1, \\ f_2 &= (l-x_2)k_2 + b_2\dot{x_2} + h_2u_2, \\ f_3 &= (2l-x_1)k_3 + b_3\dot{x_3} + h_3u_3, \\ x_3 &= x_1 + x_2, \\ \ddot{x_1} &= (f_1 - f_2 - mg)/m, \\ \ddot{x_2} &= (2f_2 + f_3 - f_1)/m \end{aligned}$$

where x_1, x_2, and x_3 are the current lengths of the sticks, f_1, f_2, and f_3 are the forces applied by the sticks due to elasticity, and the point masses m_1 and m_2 have the same weight m. Because the damping coefficients b_1, b_2, and b_3 were set to zero, the system exhibits a positive feedback effect which causes its behavior to diverge (this can be seen by setting the control signals u_1, u_2, and u_3 to zero). The constraint $x_3 = x_1 + x_2$ makes the system redundant as there are three inputs, u_1, u_2 and u_3, and only two free variables, x_1 and x_2. The parameter values are given in Table 3.

For this system, a task was defined by the desired trajectory for mass m_2 over a course of $T = 2.5$ seconds. A desired trajectory was generated as follows. First, four sine waves were generated with random frequencies ω_1, ω_2, ω_3, and ω_4, where ω_i was picked uniformly at random from the interval $[0.03, 0.3]$. The desired trajectory x_t^* was generated by $x_t^* = sin(\omega_1 t)sin(\omega_2 t) + sin(\omega_3 t)sin(\omega_4 t)$.

Constant	Value	Constant	Value	Constant	Value
k_1	20 kgm^2s^{-1}	k_2	20 kgm^2s^{-1}	k_3	40 kgm^2s^{-1}
b_1	0.0 kgm^2s^{-1}	b_2	0.0 kgm^2s^{-1}	b_3	0.0 kgm^2s^{-1}
h_1	10 $kgms^{-2}$	h_2	10 $kgms^{-2}$	h_3	20 $kgms^{-2}$
l	0.50 m	m	0.5 kg	g	10 ms^{-2}

Table 3: Values of constants used in the simulations of the spring-mass system.

Performance errors were quantified using a quadratic cost function:

$$c(x_3;x^*) = \int_0^T (x_t^* - x_{3,t})^2 dt$$

where $x_{3,t}$ is the position of mass m_2 at time t. Because this is a linear system with a quadratic cost function, the sequence of optimal (feedforward) control signals for a task can be computed using standard optimal control techniques. In our simulations, we discretized the system in time steps of size 0.025 seconds and integrated the system using a first-order Runge-Kutta method.

We created libraries of sequences as follows. We first generated a training set of 1000 tasks. For each task, we also computed the optimal sequence of control signals. Using these optimal sequences as data items, we created a library of PCA sequences by extracting the top thirty principal components based on these data items. We created a library of thirty GAR sequences using the additive regression procedure described above, with the exception that an optimal sequence (as opposed to a sequence found via feedback error learning) was added to the library when a good linear combination of library sequences could not be found.

As above, we compare the performances of four learning systems comprising all four combinations of representational primitives (PCA and GAR sequences) and optimization procedures (policy gradient and additive regression). The results on 100 test tasks are shown in Figure 11 (the leftmost bar in this figure gives the average RMSE using optimal control sequences). Note that all four learners performed nearly optimally. This is unsurprising as the quadratic error surface contains a single (global) minimum, and any reasonable optimization procedure will find this minimum. Also note that all four learners showed similar levels of performance (the differences in their performances are not statistically significant). This result is consistent with our predictions for linear systems with quadratic cost functions (see Section 5.1).

Although the learners showed similar levels of performance, a main point of this section is that they are not equivalent in terms of processing time. To quantify processing time, we examined the number of calls each learner made to the simulator of the spring-mass system. This simulator must be called each time a gradient is computed. On average, the learner using GAR sequences and additive regression made 49 calls, the learner using PCA sequences and policy gradient made 3879 calls, the learner using PCA sequences and additive regression made 62 calls, and the learner using GAR sequences and policy gradient made 3422 calls. Clearly, the additive regression algorithm is efficient in the sense that it made significantly fewer calls to the spring-mass simulator, irrespective of the library used.

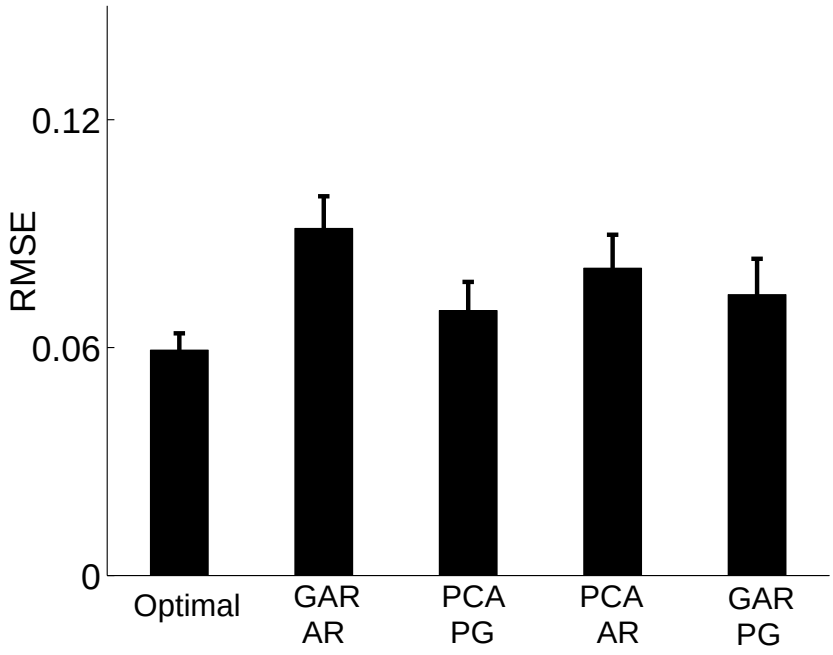


Figure 11: Results with the spring-mass system. The vertical axis shows the average RMSE on a test set with 100 tasks. The horizontal axis shows the learning system: (i) Optimal: optimal control signals; (ii) GAR+AR: GAR sequences with additive regression; (iii) PCA+PG: PCA sequences with policy gradient; (iv) PCA+AR: PCA sequences with additive regression; and (v) GAR+PG: GAR sequences with policy gradient.

7. Conclusions

In summary, the computational complexities arising in motor control can be ameliorated through the use of a library of motor synergies. We presented a new model, referred to as the Greedy Additive Regression (GAR) model, for learning a library of torque sequences, and for learning the coefficients of a linear combination of library sequences minimizing a cost function. Results using a simulated two-joint arm suggest that the GAR model consistently shows excellent performance in the sense that it rapidly learns to perform novel, complex motor tasks. Moreover, its library is overcomplete and sparse, meaning that only a small fraction of the stored torque sequences are used when learning a new movement. The library is also robust in the sense that, after an initial training period, nearly all novel movements can be learned as additive combinations of sequences in the library, and in the sense that it shows good generalization when an arm's dynamics are altered between training and test conditions, such as when a payload is added to the arm. Additionally, we showed that the GAR model works well regardless of whether motor tasks are specified in joint space or Cartesian space.

The GAR model appears to consistently outperform the PCA model, as described above. A comparison of these two models suggests why this is the case. The GAR model uses a library of local features—each sequence in its library is a solution to a single task from the training set—and a local optimization procedure, namely, additive regression. In contrast, the PCA model uses a library of global features—each item in its library reflects properties of all tasks in the training set—and policy gradient which is a global optimization procedure because it seeks good combinations of all items in its library. We conjecture that the local versus global nature of the GAR versus PCA models accounts for the performance advantages of the GAR model on nonlinear tasks. This account is consistent with other empirical findings in the machine learning literature (Friedman, 2001; Perkins, Lacker, and Theiler, 2003; Viola and Jones, 2004). Future work will need to provide a theoretical underpinning for this intuitive conjecture. The GAR and PCA models represent two ends of a local/global continuum. Future work should also study models that lie at intermediate points along this continuum, such as models that form linear combinations by adding a small number of features at each iteration, instead of the addition of a single feature as in the GAR model.

We have focused here on defining and evaluating the GAR model from a machine learning perspective. Future research will need to focus on the implications of the model for our understanding of motor control in biological organisms, the theoretical foundations of the model, and further empirical evaluations. In regard to our understanding of biological motor control, it would be interesting to know whether sets of motor synergies used by biological organisms are sparse and overcomplete as suggested by the GAR model, or are full-distributed and non-redundant as suggested by the PCA model. If they are sparse and overcomplete, then the computational advantages of the GAR model may help us understand why organisms have evolved or developed to use this type of representation. In regard to theoretical foundations, the engineering community is often reluctant to adopt new adaptive procedures for control unless these procedures have proven stability and performance guarantees. At the moment, no such guarantees exist for the GAR model. Future work will need to address these issues. In regard to empirical evaluations, future research will need to evaluate the GAR model with larger and more complex motor systems and motor tasks.

Acknowledgments

We thank two anonymous reviewers for their helpful comments on an earlier version of this manuscript. This work was supported by AFOSR research grant FA9550-06-1-0492.

References

C. G. Atkeson and D. J. Reinkensmeyer. Using associative content-addressable memories to control robots. In W. T. Miller III, R. S. Sutton, and P. J. Werbos, editors,, *Neural Networks for Control*. MIT Press, 1990.

C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75-113, 1997.

D. C. Bentivegna. *Learning from Observation Using Primitives*. Ph.D. dissertation, Georgia Institute of Technology, 2004.

N. Bernstein. *The Coordination and Regulation of Movements*. Pergamon Press, 1967.

P. Bühlmann. Boosting methods: Why they can be useful for high-dimensional data. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC)*, 2003.

M. Chhabra and R. A. Jacobs. Properties of synergies arising from a theory of optimal motor behavior. *Neural Computation*, 18:2320-2342, 2006.

P. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3:24-32, 1996.

A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature Neuroscience*, 6:300-308, 2003.

A. Fod, M. J. Matarić, and O. C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12:39-54, 2002.

Y. Freund and R. E. Schapire. A decision- theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119-139, 1997.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189-1232, 2001.

J. M. Hollerbach and T. Flash. Dynamic interactions between limb segments during planar arm movement. *Biological Cybernetics*, 44:67-77, 1982.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *Proceedings of the 21st International Conference on Machine Learning*, 2004.

M. Kawato, K. Furukawa, and R. Suzuki. Hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics*, 57:169-185, 1987.

M. Lau and J. J. Kuffner. Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2005.

J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (SIGGRAPH)*, 21:491-500, 2002.

M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12:337-365, 2000.

W. Li and E. Todorov. Iterative linear-quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the First International Conference on Informatics in Control, Automation, and Robotics*, 2004.

S. Mallat and Z. Zhang. Matching pursuit with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397-3415, 1993.

T. W. Miller III, F. H. Glanz, and L. G. Kraft. Application of a general learning algorithm to the control of robotic manipulators. *International Journal of Robotic Research*, 6:84-98, 1987.

F. A. Mussa-Ivaldi, S. F. Giszter, and E. Bizzi. Linear combination of primitives in vertebrate motor control. *Proceedings of the National Academy of Sciences USA*, 91:7534-7538, 1994.

S. Perkins, K. Lacker, and J. Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333-1356, 2003.

J. Peters and S. Schaal. Reinforcement learning for parameterized motor primitives. In *Proceedings of the International Joint Conference on Neural Networks*, 2006.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

A. Safanova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (SIGGRAPH)*, 23:514-521, 2004.

T. D. Sanger. Optimal unsupervised motor learning for dimensionality reduction of nonlinear control systems. *IEEE Transactions on Neural Networks*, 5:965-973, 1994.

T. D. Sanger. Optimal movement primitives. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*. MIT Press, 1995.

R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197-227, 1990.

E. C. Smith and M. S. Lewicki. Efficient auditory coding. *Nature*, 439:978-982, 2006.

M. Stolle and C. G. Atkeson. Policies based on trajectory libraries. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2006.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*. MIT Press, 1999.

Y. Tassa, T. Erez, and W. Smart. Receding horizon differential dynamic programming. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

K. A. Thoroughman and R. Shadmehr. Learning of action through adaptive combination of motor primitives. *Nature*, 407:742-747, 2000.

E. Todorov and Z. Ghahramani. Unsupervised learning of sensory-motor primitives. In *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2003.

E. Todorov and Z. Ghahramani. Analysis of the synergies underlying complex hand manipulation. In *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2004.

P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137-154, 2004.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229-256, 1992.

Journal of Machine Learning Research 9 (2008) 1559-1582 Submitted 10/07; Revised 2/08; Published 7/08

Aggregation of SVM Classifiers Using Sobolev Spaces

Sébastien Loustau LOUSTAU@CMI.UNIV-MRS.FR
Laboratoire d'Analyse, Topologie et Probabilités (UMR CNRS 6632)
Université Aix-Marseille 1
CMI-39 rue Joliot-Curie 13453 Marseille, France

Editor: John Shawe-Taylor

Abstract

This paper investigates statistical performances of Support Vector Machines (SVM) and considers the problem of adaptation to the margin parameter and to complexity. In particular we provide a classifier with no tuning parameter. It is a combination of SVM classifiers.

Our contribution is two-fold: (1) we propose learning rates for SVM using Sobolev spaces and build a numerically realizable aggregate that converges with same rate; (2) we present practical experiments of this method of aggregation for SVM using both Sobolev spaces and Gaussian kernels.

Keywords: classification, support vector machines, learning rates, approximation, aggregation of classifiers

1. Introduction

We consider the binary classification setting. Let $\mathcal{X} \times \{-1,1\}$ be a measurable space endowed with P an unknown probability distribution on $\mathcal{X} \times \{-1,1\}$. Let $D_n = \{(X_i,Y_i), i = 1,\dots n\}$ be n realizations of a random variable (X,Y) with law P (in the sequel we also write P_X for the marginal distribution of X). Given this training set D_n, the goal of Learning is to predict class Y of new observation X. In other words, a classification algorithm builds a decision rule from $\mathcal{X}$ to $\{-1,1\}$ or more generally a function f from $\mathcal{X}$ to $\mathbb{R}$ where the sign of $f(x)$ determines the class of an input x.

The efficiency of a classifier is measured by the *generalization error*

$$R(f) := \mathbb{P}(\text{sign}(f(X)) \neq Y),$$

where $\text{sign}(y)$ denotes the sign of $y \in \mathbb{R}$ with the convention $\text{sign}(0) = 1$. A well-known minimizer over all measurable functions of the generalization error is called the *Bayes rule*, defined by

$$f^*(x) := \text{sign}(2\eta(x) - 1)$$

where $\eta(x) := \mathbb{P}(Y = 1|X = x)$ for all $x \in \mathcal{X}$. Unfortunately, the dependence of f^* on the unknown conditional probability function η makes it uncomputable in practice.

A natural way to overcome this difficulty is to provide an empirical decision rule or classifier based on the data D_n. It has to mimic the Bayes. The way one measures the efficiency of a classifier $\hat{f}_n := \hat{f}_n(D_n)$ is via its *excess risk*:

$$R(\hat{f}_n, f^*) := R(\hat{f}_n) - R(f^*), \tag{1}$$

where here $R(\hat{f}_n) := \mathbb{P}(\text{sign}(\hat{f}_n(X)) \neq Y | D_n)$. Given P, we hence say that a classifier $\hat{f}_n$ is consistent if the expectation of (1) with respect to $P^{\otimes n}$ (the distribution of the training set) goes to zero as n goes to infinity. Finally, we can look for a way of quantifying this convergence. A classifier $\hat{f}_n$ learns with rate $(\psi_n)_{n\in\mathbb{N}}$ if there exists an absolute constant $C > 0$ such that for all integer n,

$$\mathbb{E}R(\hat{f}_n, f^*) \leq C\psi_n, \tag{2}$$

where in the sequel $\mathbb{E}$ is the expectation with respect to $P^{\otimes n}$. Of course (2) ensures consistency of $\hat{f}_n$ whenever (ψ_n) goes to zero with n.

It has been shown in Devroye (1982) that no classifier can learn with a given rate for all distributions P. However several authors propose different rates reached by restricting the class of joint distributions. Pionneering works of Vapnik (Vapnik and Chervonenkis, 1971, 1974) investigate the statistical procedure called Empirical Risk Minimization (ERM). The ERM estimator consists in searching for a classifier that minimizes the empirical risk

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{\{\text{sign}(f(X_i))\neq Y_i\}}, \tag{3}$$

over a class of prediction rules $\mathcal{F}$, where $\mathbb{1}_A$ denotes the indicator function of the set A. If we suppose that the class of decision rules $\mathcal{F}$ has finite VC dimension, ERM reaches the parametric rate $n^{-\frac{1}{2}}$ in (2) when f^* belongs to the class $\mathcal{F}$. Moreover, if P is noise-free (i.e., $R(f^*) = 0$), the rate becomes n^{-1}. This is a fast rate.

More recently, Tsybakov (2004) describes intermediate situations using a margin assumption. This assumption adds a control on the behavior of the conditional probability function η at the level $\frac{1}{2}$ (see (10) below). Under this condition, Tsybakov (2004) gets minimax fast rates of convergence for classification with ERM estimators over a class $\mathcal{F}$ with controlled complexity (in terms of entropy). These rates depend on two parameters : the margin parameter and the complexity of the class of candidates f^* (see also Massart and Nédélec, 2006). Another study of the behavior of ERM is presented in Bartlett and Mendelson (2006).

It is well known, however, that minimizing (3) is computationally intractable for many non trivial classes of functions (Arora et al., 1997). It comes from the non convexity of the functional (3). It suggests that we must use a convex surrogate Φ for the loss. The main idea is to minimize an empirical Φ-risk

$$A_n^{\Phi}(f) = \frac{1}{n}\sum_{i=1}^{n} \Phi(Y_i f(X_i)),$$

over a class $\mathcal{F}$ of real-valued functions. Then $\hat{f}_n = \text{sign}(\hat{F}_n)$ where $\hat{F}_n \in \text{Arg}\min_{f\in\mathcal{F}} A_n^{\Phi}(f)$ has a small excess risk. Recently a number of methods have been proposed, such as boosting (Freund, 1995) or Support Vector Machines. The statistical consequences of choosing a convex surrogate is well treated by Zhang (2004) and Bartlett et al. (2006). In this paper it is proposed to use the hinge loss $\Phi(v) = (1-v)_+$ (where $(\cdot)_+$ denotes the positive part) as surrogate, that is, to focus on the SVM algorithm.

SVM was first proposed by Boser et al. (1992) for pattern recognition. It consists in minimizing a regularized empirical Φ-risk over a Reproducing Kernel Hilbert Space (RKHS for short in the sequel). Given a training set D_n, the SVM optimization problem without offset can be written:

$$\min_{f\in\mathcal{H}_K} \left(\frac{1}{n}\sum_{i=1}^{n} l(Y_i, f(X_i)) + \alpha_n \| f \|_K^2 \right), \tag{4}$$

where in the sequel:

1. The functional l is called the hinge loss and is now written $l(y, f(x)) = (1 - yf(x))_+$. The first term of the minimization (4) is then the empirical Φ-risk A_n^Φ for $\Phi(v) = (1-v)_+$.

2. The space $\mathcal{H}_K$ is a RKHS with reproducing kernel K. Under some mild conditions over K, it consists of continuous functions from $\mathcal{X}$ to $\mathbb{R}$ or $\mathbb{C}$ with the reproducing property:

$$\forall f \in \mathcal{H}_K, \forall x \in \mathcal{X}, f(x) = < K(x, \cdot), f >_{\mathcal{H}_K} .$$

   Recall that every positive definite kernel has an essentially unique RKHS (Aronszajn, 1950).

3. The sequence α_n is a decreasing sequence that depends on n. This smoothing parameter has to be determined explicitly. Such a problem will be studied in this work.

4. The norm $\|.\|_K$ is the norm associated to the inner product in the Hilbert space $\mathcal{H}_K$.

For a survey on this kernel method we refer to Cristianini and Shawe-Taylor (2000).

This algorithm is at the heart of many theoretical considerations. However, its good practical performances are not yet completely understood. The study of statistical consistency of the algorithm and approximation properties of kernels can be found in Steinwart (2001) or more recently in Steinwart (2005). Blanchard et al. (2006) propose a model selection point of view for SVM. Finally, several authors provide learning rates to the Bayes for SVM (Wu and Zhou, 2006; Wu et al., 2007; Steinwart and Scovel, 2007, 2005). In these papers, both approximation power of kernels and estimation results are presented. Wu and Zhou (2006) state slow rates (logarithmic with the sample size) for SVM using a Gaussian kernel with fixed width. It holds under no margin assumption for Bayes rule with a given regularity. Steinwart and Scovel (2007) give, under a margin assumption, fast rates for SVM using a decreasing width (which depends on the sample size). An additional geometric hypothesis over the joint distribution is necessary to get a control of the approximation using Gaussian kernels.

These results focus on SVM using Gaussian kernels. The goal of this work is to clarify both practical and theoretical performances of the algorithm using two different classes of kernels. In a first theoretical part, we consider a family of kernels generating Sobolev spaces as RKHS. It gives an alternative to the extensively studied Gaussian kernels. We quantify the approximation power of these kernels. It depends on the regularity of the Bayes prediction rule in terms of Besov space. Then under the margin assumption, we give learning rates of convergence for SVM using Sobolev spaces. It holds by choosing optimally the tuning parameter α_n in (4). This choice strongly depends on the regularity assumption over the Bayes and the margin assumption. As a result, it is non-adaptive. Then we turn out into more practical considerations. Following Lecué (2007a), we give a procedure to construct directly from the data a classifier with similar statistical performances. It uses a method called aggregation with exponential weights. Finally, we show practical performances of this aggregate and compare it with a similar classifier using Gaussian kernels and results of Steinwart and Scovel (2007).

The paper is organized as follows. In Section 2, we give statistical performances of SVM using Sobolev spaces. Section 3 presents the adaptive procedure of aggregation and show the performances of the data-dependent aggregate. This procedure does not damage the learning rates stated in Section 2. We show practical experiments in Section 4 and conclude in Section 5 with a discussion. Section 6 is devoted to the proofs.

2. Statistical Performances

As a regularization procedure, minimization (4) generates two types of errors: the estimation error and the approximation error. The use of a finite sample size produces the estimation error. The approximation error can be seen as the distance between the hypothesis space and the Bayes decision rule. It comes from the use of a RKHS of continuous functions in the minimization whereas the Bayes is not continuous. The first one is random and depends on the fluctuation of the training set. The second one is deterministic and depends on the size of the RKHS. We can see coarsely that these errors are antagonist. Theorem 7 gives a choice of the regularization parameter α_n that makes the trade-off between these two errors.

For the estimation error, we will state an oracle-type inequality of the form :

$$\mathbb{E}R_l(\hat{f}_n, f^*) \leq C \inf_{f \in \mathcal{H}_K} \left(R_l(f, f^*) + \alpha_n \|f\|_K^2 \right) + \varepsilon_n, \tag{5}$$

where $R_l(f, f^*) := \mathbb{E}_P l(Y, f(X)) - \mathbb{E}_P l(Y, f^*(X))$ is the excess l-risk of f. The term ε_n must be a residual term and satisfies:

$$\varepsilon_n \leq C' \inf_{f \in \mathcal{H}_K} \left(R_l(f, f^*) + \alpha_n \|f\|_K^2 \right),$$

where $C' > 0$. Inequality (5) deals with the estimation error. It depends on the complexity of the class of functions $\mathcal{H}_K$ and the difficulty of the problem.

Hence it remains to control the infimum in the right hand side (RHS for short) of (5). Steinwart and Scovel (2007) define the approximation error function as:

$$a(\alpha_n) := \inf_{f \in \mathcal{H}_K} \left(R_l(f, f^*) + \alpha_n \|f\|_K^2 \right). \tag{6}$$

This function represents the theoretical version of the empirical minimization (4). It depends on the chosen $\mathcal{H}_K$ and the behaviour of α_n as a function of n.

Using this approach, Steinwart and Scovel (2007) study the statistical performances of SVM minimization (4) with the parametric family of Gaussian kernels. For $\sigma \in \mathbb{R}$, we define the Gaussian kernel $K_\sigma(x,y) = \exp\left(-\sigma^2 \|x-y\|^2\right)$ on the closed unit ball of $\mathbb{R}^d$ (denoted $\mathcal{X}$). The parameter σ^{-1} is called the width of the Gaussian kernel. In this paper, under a margin assumption and a geometric assumption over the distribution, they state fast learning rates for SVM. These rates hold under some specific choices of tuning parameters recalled in Sect. 4. Following Lecué (2007a), we will use this result and more precisely these choices of tuning parameters to implement the aggregate using Gaussian kernels.

2.1 Sobolev Smooth Kernels

We propose to deal with other class of kernels than the Gaussian kernels. First we need to introduce some notations. Let us consider the set of complex-valued and integrable (resp. square-integrable) functions on $\mathbb{R}^d$ denoted as $L^1(\mathbb{R}^d)$ (resp. $L^2(\mathbb{R}^d)$). On this set, we define the Fourier transform of f to be:

$$\hat{f}(\omega) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} f(t) e^{-i\omega.t} dt, \forall \omega \in \mathbb{R}^d,$$

where $x.y$ denotes the usual scalar product of $\mathbb{R}^d$ between two points $x,y \in \mathbb{R}^d$.

After the usual extension from $L^1(\mathbb{R}^d)$ to $L^2(\mathbb{R}^d)$ with Plancherel, this operator is an isometry on $L^2(\mathbb{R}^d)$. It allows us to define, for any $s \in \mathbb{R}^+$, the Sobolev space $\mathcal{W}_s^2$ (often called fractional Sobolev space) as the following subspace of $L^2(\mathbb{R}^d)$ (Malliavin, 1974):

$$\mathcal{W}_s^2 := \{ f \in L^2(\mathbb{R}^d) : \|f\|_s^2 = \int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 (1+\|\omega\|^2)^s d\omega < \infty \}. \tag{7}$$

We refer to Triebel (1992) or Adams (1975) for a large study of this well-known functional space. With such a norm, $\mathcal{W}_s^2$ is a Hilbert space endowed with the inner product defined as:

$$< f,g >_s = \int_{\mathbb{R}^d} \hat{f}(\omega)\overline{\hat{g}(\omega)}(1+\|\omega\|^2)^s d\omega,$$

where $\bar{z}$ is the complex conjugate of z in $\mathbb{C}$. Moreover it is a Hilbert space of continuous functions for any $s > \frac{d}{2}$ (due to the embedding between $\mathcal{W}_s^2$ and $C(\mathbb{R}^d)$ for any $s > \frac{d}{2}$). It can be seen as a RKHS.

In this framework, a kernel is a symmetric and positive definite function $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{C}$. For $r \in \mathbb{R}^+$, a kernel K_r will be called *Sobolev smooth kernel* with exponent $r > d$ if the associated RKHS $\mathcal{H}_{K_r}$ is such that

$$\mathcal{H}_{K_r} = \mathcal{W}_{\frac{r}{2}}^2,$$

where $\mathcal{W}_{\frac{r}{2}}^2$ is defined in (7). The restriction $r > d$ ensures that the RKHS consists of continuous functions from $\mathbb{R}^d$ to $\mathbb{C}$. Corollary 2 provides a way of constructing such a kernel.

We say that a kernel K is a *translation invariant kernel* (or RBF kernel), if for all $x,y \in \mathbb{R}^d$,

$$K(x,y) = \Phi(x-y) \tag{8}$$

for a given $\Phi : \mathbb{R}^d \mapsto \mathbb{C}$. Function Φ is often called RB function for Radial Basis function. The most popular example of translation invariant kernel is the Gaussian kernel $K_\sigma(x,y) = \exp(-\sigma^2\|x-y\|^2)$. This kernel is not a Sobolev smooth kernel (see below).

Under suitable assumptions on Φ, the following theorem gives a Fourier representation of a RKHS associated to a translation invariant kernel. The proof is given in Section 6.

Theorem 1 *Let $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{C}$ be a translation invariant kernel where in* (8) *Φ belongs to $L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$ and such that $\widehat{\Phi}$ is integrable. Then the RKHS associated to K can be written*

$$\mathcal{H}_K = \{ f \in L^2(\mathbb{R}^d) : \|f\|_K^2 = \frac{1}{(2\pi)^{d/2}} \int_S \frac{|\hat{f}(\omega)|^2}{\widehat{\Phi}(\omega)} d\omega < \infty \text{ and } \hat{f} = 0 \text{ on } \mathbb{R}^d \backslash S \}$$

with the inner product

$$< f,g >_K = \frac{1}{(2\pi)^{d/2}} \int_S \frac{\hat{f}(\omega)\overline{\hat{g}(\omega)}}{\widehat{\Phi}(\omega)} d\omega,$$

where $S := \{\omega \in \mathbb{R}^d : \widehat{\Phi}(\omega) \neq 0\}$ is the support of $\widehat{\Phi}$.

Sufficient conditions to have a Sobolev smooth kernel are:

Corollary 2 *Let K satisfying assumptions of Theorem 1. Suppose moreover that there exist constants $C, c > 0$ and a real number $s > \frac{d}{2}$ such that*

$$\widehat{\Phi}(\omega) = \frac{C}{(c + \|\omega\|^2)^s}, \forall \omega \in \mathbb{R}^d. \tag{9}$$

Then K is a Sobolev smooth kernel with exponent $r = 2s > d$.

In Section 5 we propose an example of Sobolev smooth kernel and use it into the SVM procedure.

Remark 3 (Gaussian kernels are not Sobolev smooth) *Theorem 1 can be used to define Gaussian kernels in terms of Fourier transform. Indeed, the Gaussian kernel defined above is a translation invariant kernel with RB function $\Phi(x) = \exp(-\sigma^2\|x\|^2)$. Its Fourier transform is given by*

$$\widehat{\Phi}(\omega) = \frac{1}{(\sqrt{2}\sigma)^d} \exp(-\frac{\|\omega\|^2}{4\sigma^2}).$$

Then Φ satisfies assumptions of Theorem 1. The Fourier representation of $\mathcal{H}_\sigma$ is given by:

$$\mathcal{H}_\sigma = \{ f \in L^2(\mathbb{R}^d) : \int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 \sigma^d \exp(\frac{\|\omega\|^2}{4\sigma^2}) d\omega < \infty \}.$$

From definition (7)*, it is clear that $\mathcal{H}_\sigma$ is not a Sobolev space. This integral representation of a Gaussian RKHS illustrates the smoothness of functions $f \in \mathcal{H}_\sigma$. Indeed we can see trivially that $\mathcal{H}_\sigma \subset \mathcal{H}_{K_r}$ for any fixed $\sigma, r > 0$ (because the Fourier transform of Φ is rapidly decreasing in this case). Moreover the parameter σ can be seen as a regularization parameter : the fewer is σ, the smoother are the functions in $\mathcal{H}_\sigma$. More precisely, $\sigma < \sigma'$ entails $\mathcal{H}_\sigma \subset \mathcal{H}_{\sigma'}$.*

2.2 Approximation Efficiency of Sobolev Smooth Kernels

Here we are interested in approximation properties of $\mathcal{H}_{K_r}$. We aim at bounding the approximation function $a(\alpha_n)$ defined in (6) for the procedure (4). The best case appears when $f^* \in \mathcal{H}_K$. Then we get coarsely $a(\alpha_n) \leq C\alpha_n$ where C is an absolute constant. This case is not realizable considering a continuous RKHS since the Bayes classifier is not. In this paper, we get a control of the approximation function when f^* does not belong to the RKHS. Theorem 4 provides such a result using a Sobolev smooth kernel.

Theorem 4 *Consider the approximation function $a(\alpha_n)$ defined in* (6)*, with Sobolev smooth kernel K_r such that $r > 2s > 0$. Suppose P_X satisfies $\frac{dP_X}{dx} \leq C_0$.*

Then if $f^* \in \mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$, we have:

$$a(\alpha_n) \leq C_0^{\frac{r-2s}{r-s}} \|f^*\|_{s2\infty}^{\frac{r}{r-s}} \alpha_n^{\frac{s}{r-s}},$$

where $\|.\|_{s2\infty}$ defines the norm in the Besov space $\mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$.

The proof is detailed in Section 6 where we define explicitly Besov spaces $\mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$.

Remark 5 *(*BAYES REGULARITY*) Here we get a control of the approximation function under an assumption on the smoothness of the Bayes classifier. Of course large values of s are not possible because* $f^*(x) = \text{sign}(2\eta(x)-1)$ *is not even continuous (except for the trivial case* $\eta(x) < \frac{1}{2}$ *a.s. or* $\eta(x) > \frac{1}{2}$*). More precisely, the Besov space* $\mathcal{B}^p_{s,q}(\mathbb{R}^d)$ *is included in the space of continuous functions for* $s > \frac{d}{p}$ *and* $q > 1$*. Here* $p = 2$ *then parameter s must satisfy* $s < \frac{d}{2}$ *to have* $f^* \in \mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$*. In Remark 10 we give an example of Bayes rule verifying this smoothness assumption.*

Remark 6 *(*COMPARISON WITH STEINWART AND SCOVEL, 2007*) Steinwart and Scovel (2007) propose a same type of result using Gaussian kernels. Under a geometric assumption over the distribution, they get*

$$a(\alpha_n) \leq C\alpha_n^{\frac{\gamma}{\gamma+1}},$$

where γ *is the geometric noise exponent. Here we propose a same type of result under a regularity assumption over the possible* f^**. Theorem 17 in Section 6 shows that this result can be generalized to any other kernel, using interpolation spaces.*

2.3 Learning Rates

In this work, we restrict the class of considered distributions P. We add a control on the local slope of the conditional probability function η at the level $\frac{1}{2}$. This margin assumption (we often call $|\eta - \frac{1}{2}|$ the margin) is originally due to Mammen and Tsybakov (1999) for discriminant analysis. We will use throughout this paper the following formulation: we say that P has *margin parameter* $q > 0$ if there exists a constant $c_0 > 0$ such that

$$\mathbb{P}(|2\eta(X)-1| \leq t) \leq c_0 t^q, \tag{10}$$

for all sufficiently small t.

According to Boucheron et al. (2005), this hypothesis is equivalent to the low noise or margin assumption in Tsybakov (2004). Best situation for learning appears when the conditional probability makes a jump at the level $\frac{1}{2}$. Hence (10) holds true for any positive q. It corresponds to a margin parameter $q = +\infty$, that is, $\kappa = 1$ in the sense of Tsybakov (2004).

Finally, last step of modelling consists in clipping the solution of minimization (4). For any classifier $\hat{f}$, we hence define the *clipped version* $\hat{f}^C$ with values in $[-1,1]$ by

$$\hat{f}^C(x) = \begin{cases} -1 \text{ for } x : \hat{f}(x) < -1, \\ f(x) \text{ for } x : \hat{f}(x) \in [-1,1], \\ 1 \text{ for } x : \hat{f}(x) > 1. \end{cases}$$

This operation does not modify the classification property of $\hat{f}$ since $\text{sign}(\hat{f}) = \text{sign}(\hat{f}^C)$. It produces classifiers with bounded norm $\|.\|_\infty$. It appears in several works (Bartlett, 1998; Steinwart et al., 2007). We stress that the clip does not modify the algorithm. It is done after the training as a part of the theoretical study of the algorithm. We are now on time to state the main result of this section.

Theorem 7 *Let P be a distribution over $\mathbb{R}^d \times \{-1,1\}$ such that P_X satisfies $\frac{dP_X}{dx} \leq C_0$ and* (10) *holds for $q \in [0,+\infty]$. Let $s > 0$ and suppose $f^* \in \mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$.*

Consider the SVM minimization (4) *with Sobolev smooth kernel K_r, with $r > 2s \vee d$, built on the i.i.d. sequence $(X_i,Y_i), i = 1 \dots n$ according to P.*

If we choose α_n such that

$$\alpha_n \sim n^{-\frac{r(r-s)(q+1)}{s(r(q+2)-d)+d(r-s)(q+1)}}, \tag{11}$$

then there exists a constant C which depends on r,s,d,c_0,q and C_0 such that

$$\mathbb{E}R(\hat{f}^C_n, f^*) \leq Cn^{-\gamma(q,s)},$$

where

$$\gamma(q,s) = \frac{rs(q+1)}{s(r(q+2)-d)+d(r-s)(q+1)}. \tag{12}$$

The proof of this theorem is given in Section 6.

Remark 8 *(*FAST RATES*) Rate* (12) *is a fast rate (i.e., faster than $n^{-\frac{1}{2}}$) if $\frac{rs(q+1)}{s(r(q+2)-d)+d(r-s)(q+1)} > \frac{1}{2}$. In particular, for $q = +\infty$, it corresponds to $s > \frac{rd}{r+d}$. The presence of fast rates depends on the regularity of the Bayes classifier. Unfortunately the behaviour of f^* (see Remark 4) entails $s < \frac{d}{2}$. As a result, $\frac{sr}{sr+d(r-s)} < \frac{1}{2}$ and fast rates can not be reached.*

Remark 9 *(*COMPARISON WITH STEINWART AND SCOVEL, 2007*) This theorem gives performances of SVM using a fixed kernel. On the contrary, according to Steinwart and Scovel (2007), the bandwidth of the kernel has to be chosen as a function of n. Nevertheless, rates of convergences are fast for sufficiently large geometric noise parameter. Here we cannot get fast rates for reasonable assumption over f^*.*

Remark 10 *(*OPTIMAL SMOOTHING PARAMETER*) Theorem 7 provides a particular choice of α_n to reach rates* (12)*. Other definitions for the sequence α_n give other rates of convergence. We only mention the best possible rates. It holds for a regularization parameter optimizing the statistical performances. Indeed, α_n in* (11) *makes the balance between the estimation error and the approximation error.*

Remark 11 (EXAMPLE) *Consider the one-dimensional case where $\mathcal{X} = \mathbb{R}$. Suppose f^* is such that:*

$$card\{x \in \mathbb{R} : f^* \text{ jumps at } x\} = N < \infty. \tag{13}$$

It means that the Bayes rule changes only a finite number of times over the real line. Using standard analysis, we get

$$\|f^*\|_{TV} = \int_{\mathbb{R}} |Df^*(x)|dx = 2N$$

where Df^* is the generalized derivative of f^*. Moreover, for any f, $|\hat{f}(\omega)| \leq \|f\|_{VT}/|\omega|$. Then f^* belongs to $\mathcal{W}_{s,2}$ only for $s < 1/2$. Finally, with basic properties of Besov spaces (Triebel, 1992), we have $\mathcal{W}_{s,2} = \mathcal{B}^2_{s,2} \subset \mathcal{B}^2_{s,\infty}$.

Consequently, f^* verifying (13) *belongs to $\mathcal{B}^2_{s,\infty}$ for any $s < \frac{1}{2}$. If we consider a margin parameter $q = +\infty$, we hence cannot reach the rate of convergence*

$$n^{-\frac{r}{3r-1}}$$

which corresponds to a regularity $s = \frac{1}{2}$ in the Besov space. Then the SVM using Sobolev smooth kernel H_{K_r} with $r > 1$ cannot learn with fast rate in this simple case.

3. Aggregation

Theorem 7 provides the optimal value of α_n to reach rates of convergence (12) in the context of Sobolev spaces. It holds under two ad-hoc assumptions: a margin assumption over the distribution and a regularity assumption over the Bayes rule. Hence the choice of the smoothing parameter depends on two unknown parameters: the margin parameter q and the exponent s in the Besov space. Consequently the classifier $\hat{f}_n$ of Theorem 7 cannot be constructed from the data. It is called non-adaptive.

The goal of this section is to overcome this difficulty. We propose a classifier that adapts automatically both to the margin and to regularity. In other words, we will build a decision rule from D_n which does not depend on the unknown parameters s and q. Moreover, Theorem 12 shows that this procedure of adaptation will not damage the learning rates of Theorem 7.

We use a technique called aggregation (Nemirovski, 1998; Yang, 2000). We apply the method presented in Lecué (2007a) to our framework of Sobolev smooth kernel. It consists of splitting the data into two parts : the first part in used to construct a family of classifiers. The second part is used to make a convex combination of these classifiers. We obtain an adaptive decision rule which mimics the best one over the family. Let us first describe the method.

Denote $D^1_{n_1}$ (resp. $D^2_{n_2}$) the first subsample of size n_1 (resp. second subsample of size n_2) with $n_1 + n_2 = n$. The choice of n_1 and n_2 will be discussed later. We construct a set of classifiers $(\hat{f}^{\alpha}_{n_1})_{\alpha \in \mathcal{G}(n_2)}$ defined by $\hat{f}^{\alpha}_{n_1} = sign\left(\hat{F}^{\alpha}_{n_1}\right)$ where

$$\hat{F}^{\alpha}_{n_1} := \arg\min_{f \in \mathcal{H}_{K_r}} \left(\frac{1}{n_1} \sum_{i=1}^{n_1} l(Y_i, f(X_i)) + \alpha \| f \|^2_K \right).$$

The grid $\mathcal{G}(n_2)$ is defined by

$$\mathcal{G}(n_2) := \left\{ \alpha_k = n_2^{-\phi_k} : \phi_k = \frac{1}{2} + k\Delta^{-1}, k = 0, \ldots, \lfloor \frac{(2r-d)\Delta}{2d} \rfloor \right\},$$

with $\Delta = n_2^b$ for some $b > 0$. We hence have $\left\lfloor \frac{(2r-d)\Delta}{2d} \right\rfloor + 1$ classifiers to aggregate.

The procedure of aggregation uses the second subsample $D^2_{n_2}$ to construct a convex combination with exponential weights. Namely, the aggregate $\tilde{f}_n$ is defined by

$$\tilde{f}_n = \sum_{\alpha \in \mathcal{G}(n_2)} \omega^{(n)}_{\alpha} \hat{f}^{\alpha}_{n_1}, \tag{14}$$

where

$$\omega^{(n)}_{\alpha} = \frac{\exp\left(\sum_{i=n_1+1}^{n} Y_i \hat{f}^{\alpha}_{n_1}(X_i)\right)}{\sum_{\alpha' \in \mathcal{G}(n_2)} \exp\left(\sum_{i=n_1+1}^{n} Y_i \hat{f}^{\alpha'}_{n_1}(X_i)\right)}.$$

We hence have the following result.

Theorem 12 *Consider the classifier $\tilde{f}_n$ defined in* (14) *where $n_2 = \lceil a\frac{n}{\log n} \rceil$ for $a > 0$. Let K a compact of $(0,\infty)^2$. Then there exists a constant C which depends on r,d,c_0,K,a,b,L and C_0 such that for all $(q,s) \in K$*

$$\sup_{P \in Q_{q,s}} \mathbb{E}R(\tilde{f}_n, f^*) \leq Cn^{-\gamma(q,s)},$$

where

$$\gamma(q,s) = \frac{rs(q+1)}{s(r(q+2)-d)+d(r-s)(q+1)}$$

and $Q_{q,s}$ is the set of distributions P satisfying $\frac{dP_X}{dx} < C_0$, (10) *with parameter q and such that $f^* \in \mathcal{B}^2_{s,\infty}(\mathbb{R}^d, L) = \{f \in \mathcal{B}^2_{s,\infty}(\mathbb{R}^d) : \|f\| \leq L\}$.*

Remark 13 *Same rates as in Theorem 7 are attained. Here we deal with an implementable classifier. In Section 5 we sum up practical performances of this aggregate.*

Remark 14 *Instead of aggregating a power of n classifiers, only $\log n$ classifiers are enough to obtain this result. Lecué (2007b) states an oracle inequality such as* (22) *without any restriction on the number of estimators to aggregate.*

Remark 15 *(*AVERAGE OF AGGREGATES*) This method supposes, for a given n_1 and n_2, an arbitrary choice for the subsample $D^1_{n_1}$ and $D^2_{n_2}$. However we can use different splits of the training set. We get an average of aggregates, namely*

$$\overline{f}_n = \frac{1}{M}\sum_{k=1}^{M} \tilde{f}_n^k.$$

It does not depend on a particular split. Each $\tilde{f}_n^k$ is defined in (14) *for the split number k. With (Lecué, 2007a, Theorem 2.4), this average satisfies the oracle inequality* (22)*. Then Theorem 12 holds for $\overline{f}_n$ for any family of M splits, for $M \leq C_n^{n_1}$.*

4. Practical Experiments

We now propose experiments illustrating performances of the aggregate of Section 3. We study SVM classifiers using both Sobolev spaces and Gaussian kernels. The aggregates were implemented in **R** using the free library *kernlab*. It contains implementations of support vector machines. For a description of this package for kernel-based learning methods in **R**, we refer to Karatzoglou et al. (2007). We use real world data sets from benchmark repository[1] used by Rätsch et al. (1998). We consider 9 data sets called "Banana", "Titanic", "Thyroid", "Diabetes", "Breast-Cancer", "Flare-solar", "Heart", "Image" and "Waveform". These data sets are explained in Table 1. For each data set, we have several realizations of training and test set. The dimension of the input space is denoted by d whereas the number of observations for the training set is n. It follows the notations used in the previous sections. On each realization, we train and test our classifiers. The results presented in Table 2,3,4 show the average test errors over these realizations and the standard deviations.

1. Data sets are available online at this address http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm.

Data Set	d	n	test sample	realizations
Banana	2	400	4900	100
Titanic	3	150	2051	100
Thyroid	5	140	75	100
Diabetis	8	468	300	100
Breast-cancer	9	200	77	100
Flare-solar	9	666	400	100
Heart	13	170	100	100
Image	18	1300	1010	20
Waveform	21	400	4600	100

Table 1: Description of the data sets

4.1 SVM Using Sobolev Smooth Kernel

The first step is to pick up a Sobolev smooth kernel. Consider the following class of RBF kernels, with Radial Basis function Φ:

$$K(x,y) = \Phi(x-y) = \exp\left(-\sigma\|x-y\|\right), \forall \sigma \in \mathbb{R}. \tag{15}$$

For a given σ, this kernel is called a Laplacian kernel. It is clear that $\Phi \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$. Recall the Fourier transform of $\Phi : \mathbb{R}^d \mapsto \mathbb{R}$ (see Williamson et al., 2001):

$$\widehat{\Phi}(\omega) = 2^{\frac{d}{2}} \pi^{-\frac{1}{2}} \Gamma(\frac{d}{2}+1) \frac{\sigma}{(\sigma^2 + \|\omega\|^2)^{\frac{d+1}{2}}}, \forall \omega \in \mathbb{R}^d,$$

where $\Gamma(x) = \int_{\mathbb{R}^+} e^{-t} t^{x-1} dt$ is the Gamma function.

With Corollary 2, for any fixed σ, the Laplacian kernel defined in (15) is a Sobolev smooth kernel with exponent $r = d+1$. It satisfies assumptions of Theorem 7 and can be used in the implementation of the algorithm.

It is worth noticing that the parameter σ is constant. If we take a significantly small value for σ, as $\sigma = n^{-u}$, $u > 0$, (9) holds for C and c depending on n. Thus Corollary 2 does not hold. To avoid this problem, we choose in our aggregation step using this class of kernels a constant $\sigma = 5$. In the sequel the Laplacian kernel used is precisely $K(x,y) = \exp(-5\|x-y\|)$.

Table 2 shows the first experiments. For each realization of training set, we use previous section to build

- the set of classifiers $(\hat{f}_{n_1}^{\alpha})$ for α belonging to $\mathcal{G}(n_2)$;
- exponential weights $\omega_{\alpha}^{(n)}$ to deduce aggregate $\tilde{f}_n$.

Recall the definition of $\mathcal{G}(n_2)$ in this case:

$$\mathcal{G}(n_2) := \left\{ \alpha_k = n_2^{-\phi_k} : \phi_k = \frac{1}{2} + k\Delta^{-1}, k = 0, \ldots, \lfloor \frac{(2r-d)\Delta}{2d} \rfloor \right\},$$

where $\Delta = n_2^b$. We take $b = 1$ in the construction. Instead of a step $\Delta = n_2^b$, it is possible to take only $\Delta = \log n_2$ (see Remark 14) . The value of b governs the size of the grid. The cardinal is

given in Table 2 for each data set. Note that growing b does not improve significantly the performances whereas it adds computing time. Indeed, whatever b, $\mathcal{G}(n_2)$ is contained in this case into $[n_2^{-\frac{d+1}{d}}, n_2^{-\frac{1}{2}}]$. This location is motivated by Theorem 7, namely equation (11).The value of b only deals with the distance between each point of $\mathcal{G}(n_2)$. It does not change the location of the grid.

Table 2 relates the average test errors and the standard deviations. We first collect the performances of the family of weak estimators $(\hat{f}_{n_1}^{\alpha}), \alpha \in \mathcal{G}(n_2)$. We mention in order the performances of the worst estimator, the mean over the family and the best over the family. It gives an idea of the estimators to aggregate. Then the performances of the aggregate using exponential weights are given in the last column.

Data Set	card$\mathcal{G}(n_2)$	max	mean	min	Laplace Aggregate
Banana	102	11.41±0.58	11.33± 0.57	11.12±0.59	11.31± 0.57
Titanic	38	22.80±1.16	22.80±1.14	22.77±1.13	22.77±1.13
Thyroid	31	5.97±2.61	5.45±2.56	4.77±2.63	5.45±2.68
Diabetis	72	29.56±2.03	28.40±2.00	27.33±1.96	28.34±2.27
Breast-cancer	35	35.10±5.34	33.26±5.06	31.49±5.05	32.74±5.16
Flare-solar	95	35.97±1.94	35.68±1.90	35.52±1.90	35.69±1.93
Heart	29	22.38±3.97	22.11±3.98	21.76±3.99	22.12±3.98
Image	152	4.35±0.87	4.06±0.74	3.79±0.74	3.95±0.74
Waveform	56	14.51±0.70	14.16±0.67	13.78±0.65	14.12±0.72

Table 2: Performances using Laplacian kernel

Note that the amplitude in the family is not very important. It may be explain by its construction. Indeed, $\mathcal{G}(n_2)$ is motivated by Theorem 7, which gives the location of the grid (see above). This family has a mathematical justification. The test errors of the aggregate are located between the average over the family and the oracle of the family.

A temperature parameter usually appears in aggregation methods. It governs the variations of values $\omega_{\alpha}^{(n)}$, for $\alpha \in \mathcal{G}(n_2)$. In Table 2 the weak classifiers have almost the same performances. This could explain why no temperature parameter is needed here.

4.2 SVM Using Gaussian Kernels

Here we focus on the parametric class of Gaussian kernels $K_\sigma(x,y) = \exp\left(-\sigma^2\|x-y\|^2\right)$, for $\sigma \in \mathbb{R}$. We build an aggregate made of a convex combination of Gaussian SVM classifiers. In this case, the construction is not exactly the same. It comes from Steinwart and Scovel (2007). In this paper, they introduce a geometric noise assumption. This hypothesis deals with the concentration of the measure $|2\eta - 1|P_X$ near the decision boundary. It allows to control the approximation function (6). According to Steinwart and Scovel (2007), suppose that the probability distribution P has a geometric noise $\gamma > 0$ and assumption (10) holds with margin parameter $q > 0$. Then if we choose

$$\alpha_n = \begin{cases} n^{-\frac{\gamma+1}{2\gamma+1}} \text{ if } \gamma \leq \frac{q+2}{2q}, \\ n^{-\frac{2(\gamma+1)(q+1)}{2\gamma(q+2)+3q+4}}, \text{ otherwise} \end{cases}$$

the solution of (4) using a Gaussian kernel K_σ with $\sigma = \alpha_n^{-\frac{1}{(\gamma+1)d}}$ learns with rates

$$\begin{cases} n^{-\frac{\gamma}{2\gamma+1}+\varepsilon} \text{ if } \gamma \leq \frac{q+2}{2q}, \\ n^{-\frac{2\gamma(q+1)}{2\gamma(q+2)+3q+4}+\varepsilon} \text{ otherwise,} \end{cases}$$

for all $\varepsilon > 0$.

We can see that the variance of the Gaussian kernels is not fixed. It has to be chosen as a function of the geometric noise exponent. As a result, parameter σ must be considered in the aggregation procedure, as the smoothing parameter α. It gives a two-dimensional grid of Gaussian SVM of the following form (Lecué, 2007a):

$$\mathcal{N}(n_2) = \left\{ (\sigma_{n_2,\phi}, \alpha_{n_2,\psi}) = (n_2^{\phi/d}, n_2^{-\psi}) : (\phi,\psi) \in \mathcal{M}(n_2) \right\}$$

where

$$\mathcal{M}(n_2) = \left\{ (\phi_{n_2,p_1}, \psi_{n_2,p_2}) = \left(\frac{p_1}{2\Delta}, \frac{p_2}{\Delta} + \frac{1}{2} \right) : p_1 = 1,\dots,2\lfloor\Delta\rfloor; p_2 = 1,\dots,\lfloor\Delta/2\rfloor \right\},$$

for $\Delta = n_2^b$. Thus we have more classifiers to aggregate and needs more time to run. As a consequence, we choose constant $b = 0.5$ in our experiments. Such as the Sobolev case, the number of classifiers to aggregate is mentioned in Table 3 for each data set.

Table 3 relates the generalization performances of the classifiers over the test samples. We first give the performances of the family of Gaussian SVM (namely the worst, the mean and the oracle over the family). The performances of the aggregate using exponential weights are given in the last column.

Data Set	card$\mathcal{N}(n_2)$	max	mean	min	gaussian aggregate
Banana	100	17.29± 3.08	12.27±0.89	10.85±0.63	11.43±0.84
Titanic	36	23.15±1.30	22.81±1.00	22.49±0.78	22.57±0.79
Thyroid	36	8.19±2.63	6.76±2.72	5.59±2.94	6.31±2.97
Diabetis	100	29.82±1.98	28.19±1.84	26.39±1.85	27.80±2.06
Breast-cancer	42	34.83±5.12	32.76±4.82	30.48±4.61	32.13±4.77
Flare-solar	144	39.06±1.92	36.01±1.54	34.09±1.69	34.87±1.82
Heart	42	23.1±3.80	22.60±3.71	21.99±3.59	22.62±3.77
Image	256	7.79±1.00	6.33±0.83	5.30±0.73	5.66±0.74
Waveform	100	15.41±0.80	15.08±0.78	14.72±0.77	15.04±0.79

Table 3: Performances using Gaussian kernels

In this case the generalization errors in the family are more disparate. It comes from a two-dimensional grid of parameters. The performances of the Gaussian aggregate, as above, are located between the average of weak estimators and the best among the family.

4.3 Comparison With Rätsch et al. (1998)

Table 4 combines the performances of the aggregates using Laplacian kernel and Gaussian kernels. The errors are comparable. Gaussian kernels and Laplacian kernel lead to similar performances. Then we mention the generalization errors of Rätsch et al. (1998).

Rätsch et al. (1998) proposes generalizations of the original Adaboost algorithm. However, extensive simulations are presented like experimental results for SVM using Gaussian kernels. The choice of the parameters (α_n, σ) are done by 5-fold-cross validation thanks to several training data sets. This approach has not any mathematical justification. Moreover their mathematical programming problems are distributed over 30 computers. We only use last column to have an idea of reasonable average test errors for these data sets.

Data Set	Laplace Aggregate	Gaussian Aggregate	Rätsch et al. (1998)
Banana	11.31± 0.57	11.43±0.84	11.53±0.66
Titanic	22.77±1.13	22.57±0.79	22.42±1.02
Thyroid	5.45±2.68	6.31±2.97	4.80±2.19
Diabetis	28.34±2.27	27.80±2.06	23.53±1.76
Breast-cancer	32.74±5.16	32.13±4.77	26.04±4.74
Flare-solar	35.69±1.93	34.87±1.82	32.43±1.82
Heart	22.12±3.98	22.62±3.77	15.95±3.26
Image	3.95±0.74	5.66±0.74	2.96±0.6
Waveform	14.12±0.72	15.04±0.79	9.88±0.83

Table 4: Comparison with Rätsch et al. (1998).

Table 4 illustrates good resistance of our aggregates when the dimension is not too large. Nevertheless, in the last columns, our estimators fail. This may have a theoretical explanation. In Theorem 7 and 12, a constant C appears in the upper bounds. This constant in front of the rates of convergence depends on the dimension of the input space. Increasing d grows this constant C and may affect the performances. Moreover, the choice of the parameters in Rätsch et al. (1998) are done with several training sets. In our approach, for each realization of a training set, we construct an adaptive classifier using n observations. The amount of information used is not the same. It may also explain this difference.

5. Conclusion

This paper gives some insights into SVM algorithm, from both theoretical and practical point of view. We have tackled several important questions such as its statistical performances, the role of the kernel and the choice of the tuning parameters.

The first part of the paper focuses on the statistical performances of the method. In this study, we consider Sobolev smooth kernels as an alternative to the Gaussian kernels. It allows us to bring out a functional class of Bayes rule (namely Besov spaces $\mathcal{B}^2_{s,\infty}$) ensuring good approximation properties for our hypothesis space. Explicit rates of convergence have been given depending on the margin and the regularity (Theorem 7). Nevertheless, this result was non-adaptive.

Then it has been necessary to consider the problem of adaptation. The aggregation method appeared suitable in this context to construct directly from the data a competitive decision rule: it has the same statistical performances as the non-adaptive classifier (Theorem 12). In this procedure, we use explicitly the theoretical part to choose the scale of tuning parameters. For completeness, we have finally implemented the method and gave practical performances over real benchmark data sets. These practical experiments are to be considered as preliminary. However it shows similar

performances for SVM using Gaussian or non-Gaussian kernel. Moreover it illustrates rather well the importance of constructing a classifier with some mathematical background.

6. Proofs

This section contains proofs of the results presented in this paper.

6.1 Proof of Theorem 1 and Corollary 2

We consider a translation invariant kernel $K : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{C}$ with RB function Φ satisfying assumptions of Theorem 1. The following lemma will be useful.

Lemma 16 *For any $y \in \mathbb{R}^d$, consider the function $k_y : x \mapsto K(x,y)$ defined in $\mathbb{R}^d$. Then we have the following statements:*

1. $k_y(x) = \overline{\hat{g}_y(x)}$ *where* $g_y(\omega) = e^{i\omega.y}\widehat{\Phi}(\omega)$.
2. $\hat{k}_y(\omega) = e^{-i\omega.y}\widehat{\Phi}(\omega)$.

Proof

1. $\Phi \in L^2(\mathbb{R}^d)$ hence the inverse Fourier formula allows us to write :

$$\begin{aligned} k_y(x) = \Phi(x-y) &= \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{i\omega.(x-y)} \widehat{\Phi}(\omega) d\omega \\ &= \frac{1}{(2\pi)^{d/2}} \overline{\int_{\mathbb{R}^d} e^{-i\omega.x} e^{i\omega.y} \widehat{\Phi}(\omega) d\omega} \\ &= \frac{1}{(2\pi)^{d/2}} \overline{\int_{\mathbb{R}^d} e^{-i\omega.x} g_y(\omega) d\omega}. \end{aligned}$$

2. Now using 1. one gets

$$\hat{k}_y(\omega) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{-i\omega.x} k_y(x) dx = \frac{1}{(2\pi)^{d/2}} \overline{\int_{\mathbb{R}^d} e^{i\omega.x} \hat{g}_y(x) dx}.$$

Gathering with the inverse Fourier transform of $g_y \in L^2(\mathbb{R}^d)$, we have

$$\hat{k}_y(\omega) = \overline{g_y(\omega)} = e^{-i\omega.y}\widehat{\Phi}(\omega).$$

■

Proof (of Theorem 1)

We write

$$\mathcal{H}_0 = \{ f \in L^2(\mathbb{R}^d) : \int_S \frac{|\hat{f}(\omega)|^2}{\widehat{\Phi}(\omega)} d\omega < \infty \text{ and } \hat{f} = 0 \text{ on } S \},$$

with the corresponding norm

$$\|f\|_{\mathcal{H}_0} := \sqrt{\frac{1}{(2\pi)^{d/2}} \int_S \frac{|\hat{f}(\omega)|^2}{\widehat{\Phi}(\omega)} d\omega}.$$

We will show that $\mathcal{H}_0$ coincides with $\mathcal{H}_K$.

For a given $y \in \mathbb{R}^d$, from Lemma 16 it is clear that $\hat{k}_y(\omega) = 0$ for $\omega \in \mathbb{R}^d \backslash S$. Moreover using again Lemma 16:

$$\int_S \frac{|\hat{k}_y(\omega)|^2}{\widehat{\Phi}(\omega)} d\omega = \int_S \widehat{\Phi}(\omega) d\omega < \infty$$

since $\widehat{\Phi}$ is integrable. Then $k_y \in \mathcal{H}_0$ for any $y \in \mathbb{R}^d$. Now we have to establish that $\mathcal{H}_0$ is a Hilbert space. Following Matache and Matache (2002), we can show that, for any $f \in \mathcal{H}_0$:

$$\|\hat{f}\|_1 \leq \sqrt{(2\pi)^{d/2} \|\widehat{\Phi}\|_1} \|f\|_{\mathcal{H}_0} \text{ and } \|\hat{f}\|_2 \leq \sqrt{(2\pi)^{d/2} \|\Phi\|_1} \|f\|_{\mathcal{H}_0},$$

where $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the norms in $L^1(\mathbb{R}^d)$ and $L^2(\mathbb{R}^d)$.

Indeed, by Cauchy-Schwarz,

$$\int_{\mathbb{R}^d} |\hat{f}(\omega)| d\omega \leq \sqrt{\frac{1}{(2\pi)^{d/2}} \int_S \frac{|\hat{f}(\omega)|^2}{\widehat{\Phi}(\omega)} d\omega} \sqrt{(2\pi)^{d/2} \int_S \widehat{\Phi}(\omega) d\omega}.$$

Moreover, since $\|\widehat{\Phi}\|_\infty \leq \|\Phi\|_1$,

$$\int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 d\omega \leq \|\Phi\|_1 \int_S \frac{|\hat{f}(\omega|^2}{\widehat{\Phi}(\omega)} d\omega.$$

Then considering a Cauchy sequence $(f_n)_{n\in\mathbb{N}}$ in $\mathcal{H}_0$ endowed with $\|\cdot\|_{\mathcal{H}_0}$, $(\hat{f}_n)_{n\in\mathbb{N}}$ will be a Cauchy sequence in both $L^1(\mathbb{R}^d)$ and $L^2(\mathbb{R}^d)$. We conclude with Matache and Matache (2002) that $(f_n)_n$ is convergent in $\mathcal{H}_0$. Then $\mathcal{H}_0$ is complete and becomes a Hilbert space endowed with the following inner product:

$$< f, g >_{\mathcal{H}_0} = \frac{1}{(2\pi)^{d/2}} \int_S \frac{\hat{f}(\omega)\overline{\hat{g}(\omega)}}{\widehat{\Phi}(\omega)} d\omega.$$

Finally reproducing property holds. Indeed let $f \in \mathcal{H}_0$. Using again Lemma 16 :

$$f(x) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} e^{i\omega.x} \hat{f}(\omega) d\omega = \frac{1}{(2\pi)^{d/2}} \int_S \frac{\hat{f}(\omega)}{\widehat{\Phi}(\omega)} \overline{\hat{k}_x(\omega)} d\omega = < f, k_x >_{\mathcal{H}_0} .$$

We have already shown that $\forall x \in \mathbb{R}^d$, $k_x \in \mathcal{H}_0$. As a result, the unicity of the RKHS for a given kernel concludes the proof. ■

Proof (of Corollary 2)

First we have trivially that $\widehat{\Phi}$ is integrable since $s > \frac{1}{2}$. We can hence apply Theorem 1 to have

$$\mathcal{H}_K = \{ f \in L^2(\mathbb{R}^d) : \int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 (c + \|\omega\|^2)^s d\omega < \infty \},$$

since the support of $\widehat{\Phi}$ is $\mathbb{R}^d$. This expression of the RKHS associated to K corresponds, up to a constant, to the Sobolev space $\mathcal{W}_s^2$ defined in (7). Then K is a Sobolev smooth kernel with exponent $r = 2s$. ■

6.2 Proof of Theorem 4

First introduce the notion of interpolation space (Bennett and Sharpley, 1988). We restrict ourselves to a description of the real interpolation method. Let $(B, \|.\|_B)$ be a Banach space and $\mathcal{H}$ a Hilbert space dense in B. The Peetre's functional for the couple $(B, \mathcal{H})$ is defined by, for $t > 0$,

$$P(f, t, B, \mathcal{H}) := \inf \left\{ \|f_0\|_B + t \|f_1\|_{\mathcal{H}}, f = f_0 + f_1 \text{ such that } f_0 \in B, f_1 \in \mathcal{H} \right\}.$$

For fixed $t > 0$, the functional P defines a norm in the Banach space B. It is therefore a simple way to define the interpolation space between B and $\mathcal{H}$ entirely in terms of this functional. Given $\theta \in]0,1[$ and $q \in [0, \infty]$, the space $(B, \mathcal{H})_{\theta,q}$ called *interpolation space between B and* $\mathcal{H}$ consists of all $f \in B$ such that

$$\|f\|_{\theta,q} := \begin{cases} \left(\int_0^{+\infty} t^{-\theta q} P(f, t, B, \mathcal{H})^q \frac{dt}{t} \right)^{\frac{1}{q}} \text{ if } q < \infty, \\ \sup_{t>0} \left\{ t^{-\theta} P(f, t, B, \mathcal{H}) \right\} \text{ if } q = \infty \end{cases}$$

is finite.

Here we are interested in the case $q = \infty$ and the following geometric explanation of interpolation space (Smale and Zhou, 2003, Theorem 3.1):

$$f \in (B, \mathcal{H})_{\theta,\infty} \Longrightarrow \inf_{g \in B_{\mathcal{H}}(R)} \|f - g\|_B \leq \|f\|_{\theta,\infty}^{\frac{1}{1-\theta}} \left(\frac{1}{R} \right)^{\frac{\theta}{1-\theta}}, \tag{16}$$

where $B_{\mathcal{H}}(R) := \left\{ f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq R \right\}$. Hence the interpolation space between $\mathcal{B}$ and $\mathcal{H}$ satisfies $\mathcal{H} \subset (\mathcal{B}, \mathcal{H})_{\theta,\infty} \subset \mathcal{B}$. To be more precise it consists of functions located at a polynomial decreasing distance in $\mathcal{B}$ from a ball in $\mathcal{H}$ of radius R as a function of R. It would be useful to control the approximation error function in our framework.

Theorem 17 *Consider $a(\alpha_n)$ defined in* (6). *Suppose the marginal of X is such that $\frac{dP_X}{dx} \leq C_0$. Then if $f^* \in (L^2(\mathbb{R}^d), \mathcal{H}_K)_{\theta,\infty}$ we have:*

$$a(\alpha_n) \leq \|f^*\|_{\theta,\infty}^{\frac{2}{2-\theta}} \alpha_n^{\frac{\theta}{2-\theta}}.$$

Proof By the lipschitz property of the hinge loss, we have clearly since $\frac{dP_X}{dx} \leq C_0$:

$$\begin{aligned} a(\alpha_n) &\leq \inf_{f \in \mathcal{H}_K} \left(\|f - f^*\|_{L^1(P_X)} + \alpha_n \|f\|_K^2 \right) \\ &\leq \inf_{R>0} \left(C_0 \inf_{f \in B_{\mathcal{H}_K}(R)} \|f - f^*\|_{L^2(\mathbb{R}^d)} + \alpha_n R^2 \right). \end{aligned}$$

Now from (16), it follows that if $f^* \in (L^2(\mathbb{R}^d), \mathcal{H}_K)_{\theta,\infty}$,

$$a(\alpha_n) \leq \inf_{R>0} \left(\|f^*\|_{\theta,\infty}^{\frac{1}{1-\theta}} \left(\frac{1}{R}\right)^{\frac{\theta}{1-\theta}} + \alpha_n R^2 \right).$$

Optimizing with respect to R leads to the conclusion. ■

Let introduce Besov spaces $\mathcal{B}^p_{s,q}(\mathbb{R}^d)$. A Besov space is a collection of functions with common smoothness, in terms of modulus of continuity. This is a large class of functional spaces, including in particular the Sobolev spaces defined in (7) ($\mathcal{W}^2_s = \mathcal{B}^2_{s,2}(\mathbb{R}^d)$ for any $s>0$) and the Hölder spaces ($H^s = \mathcal{B}^s_{\infty,\infty}(\mathbb{R}^d)$ for any $s>0$). For a large study, we refer to Triebel (1992).

Here we restrict ourselves to the spaces $\mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$. For any $h \in \mathbb{R}^d$, we write I for the identity operator, T_h for the translation operator ($T_h(f,x) = f(x+h)$) and $\Delta_h^r := (T_h - I)^r$ for the difference operator. The modulus of continuity of order r of a function $f \in L^2(\mathbb{R}^d)$ is then

$$\omega_r(f,t)_2 = \sup_{|h|\leq t} \|\Delta_h^r(f)\|_{L^2(\mathbb{R}^d)}.$$

Then the Besov space $\mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$ consists of all functions f such that the semi-norm

$$\|f\|_{s,\infty} = \sup_{t>0} t^{-s}\omega_r(f,t)_2$$

is finite.

If we add $\|f\|_{L^2(\mathbb{R}^d)}$ to this semi-norm, we obtain the usual norm of $\mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$.

Lemma 18 *Let $s>0$ and $0<\theta<1$. Then,*

$$(L^2(\mathbb{R}^d), \mathcal{W}^2_s)_{\theta,\infty} = \mathcal{B}^2_{\theta s,\infty}(\mathbb{R}^d).$$

A proof is presented by Triebel (1978) in a more general framework.

Proof (of Theorem 4)

From the definition of Sobolev smooth kernels, we have $\mathcal{H}_{K_r} = \mathcal{W}^2_{\frac{r}{2}}$. Hence we obtain with Lemma 18:

$$(L^2(\mathbb{R}^d), \mathcal{H}_{K_r})_{\theta,\infty} = \mathcal{B}^2_{\frac{\theta r}{2},\infty}(\mathbb{R}^d).$$

Applying Theorem 17 with $\theta = \frac{2s}{r}$, this ends up the proof since P_X satisfies $\frac{dP_X}{dx} < C_0$. ■

6.3 Proof of Theorem 7

In order to control the generalization error, we have to state an inequality such as (5). We propose to use a stochastic oracle inequality from Steinwart et al. (2007). This result takes place under a margin assumption of the type (10) and a complexity assumption over the used RKHS.

We define the covering numbers of a subset A of a Banach space (E,d) as :

$$\mathcal{N}(A,\varepsilon,E) = \min\{n \geq 1 : \exists x_1, \dots x_n \in E \text{ such that } A \subset \cup_{i=1}^n B_d(x_i,\varepsilon)\}.$$

Furthermore, given a realization $T = \{(x_1, y_1), \dots (x_n, y_n)\}$ of the training set, we denote by $L^2(T_X)$ the space of all equivalence classes of functions $f : \mathcal{X} \mapsto \mathbb{R}$ such that the norm

$$\|f\|_{L^2(T_X)} := \left(\frac{1}{n} \sum_{i=1}^{n} f(x_i)^2 \right)^{1/2} \tag{17}$$

is finite. Then we can consider the behaviour of $\log \mathcal{N}(B_{\mathcal{H}_K}, \varepsilon, L^2(T_X))$ as a complexity measure for the used RKHS.

Proposition 19 (Steinwart and Scovel, 2007) *Let P be a distribution on $\mathcal{X} \times \{-1, 1\}$ and $\mathcal{H}_K$ a RKHS of continuous functions on $\mathcal{X}$. Suppose*

1. *There exists $q \in [0, +\infty]$ and $c_0 > 0$ such that*

$$\mathbb{P}(|2\eta(X) - 1| \leq t) \leq c_0 t^q, , \forall t > 0.$$

2. *There exist $a \geq 1, 0 < p < 1$ such that*

$$\sup_{T \in (\mathcal{X} \times \mathcal{Y})^n} \log \mathcal{N}\left(B_{\mathcal{H}_K}, \varepsilon, L^2(T_X)\right) \leq a \varepsilon^{-2p}, \forall \varepsilon > 0. \tag{18}$$

Then there exist constants $c \geq 1$, $\kappa, \kappa', \kappa'' > 0$ such that for all $x \geq 1$, the clipped version $\hat{f}_n^C$ of SVM classifier $\hat{f}_n$ satisfies, with probability larger than $1 - e^{-x}$,

$$\begin{aligned} R_l(\hat{f}_n^C, f^*) &\leq c \inf_{f \in \mathcal{H}_K} \left(\mathbb{E}_P (l(f) - l(f^*)) + \alpha_n \|f\|_K^2 \right) + \frac{\kappa}{n \alpha_n^p} \\ &+ \left(\frac{\kappa}{n \alpha_n^p} \right)^{\frac{q+1}{q+2-p}} + \frac{\kappa'}{n^{\frac{q+1}{q+2}}} + \frac{\kappa'' x}{n}. \end{aligned}$$

Proof (of Theorem 7)

The hinge loss $l(y, f(x)) = (1 - y f(x))_+$ satisfies, for all classifier $\hat{f}$ (Zhang, 2004):

$$R(\hat{f}, f^*) \leq R_l(\hat{f}, f^*). \tag{19}$$

Therefore, to control the excess risk of a classifier, it is sufficient to control the RHS of (19).

We apply Proposition 19 for the stochastic part and Theorem 4 for the approximation part of the analysis.

Recall a standard result for covering numbers of Sobolev spaces (Chen et al., 2004):

$$\log \mathcal{N}(B_{\mathcal{W}_r^2}, \varepsilon, C(\mathbb{R}^d)) \leq a \varepsilon^{-\frac{d}{r}}, \tag{20}$$

where constant $a := a(d)$. From (17) we have $\|f\|_{L^2(T_X)} \leq \|f\|_\infty$ for any $f \in C(\mathbb{R}^d)$, $T \in (\mathcal{X} \times \mathcal{Y})^n$. Then (20) holds true for $\log \mathcal{N}(B_{\mathcal{W}_r^2}, \varepsilon, L^2(T_X))$ uniformly over $T \in (\mathcal{X} \times \mathcal{Y})^n$. Gathering with $\mathcal{H}_{K_r} = \mathcal{W}_{r/2}^2$, the RKHS $\mathcal{H}_{K_r}$ satisfies (18) of Proposition 19 with $p = \frac{d}{r}$. Applying Proposition 19, there exist $c \geq 1$, $\kappa, \kappa', \kappa'' > 0$ such that, for all $x \geq 1$, with probability larger than $1 - e^{-x}$,

$$\begin{aligned} R_l(\hat{f}_n^C, f^*) &\leq c \inf_{f \in \mathcal{H}_K} \left(R_l(f, f^*) + \alpha_n \|f\|_K^2 \right) + \frac{\kappa}{n \alpha_n^{\frac{d}{r}}} \\ &+ \left(\frac{\kappa}{n \alpha_n^{\frac{d}{r}}} \right)^{\frac{q+1}{q+2-d/r}} + \frac{\kappa'}{n^{\frac{q+1}{q+2}}} + \frac{\kappa'' x}{n}. \end{aligned}$$

Since $f^* \in \mathcal{B}^2_{s,\infty}(\mathbb{R}^d)$, we get from Theorem 4 that with probability larger than $1-e^{-x}$,

$$R_l(\hat{f}_n^C, f^*) \leq cC_0^{\frac{r}{r-s}} \|f^*\|_{s,\infty}^{\frac{r}{r-s}} \alpha_n^{\frac{s}{r-s}} + \frac{\kappa}{n\alpha_n^{\frac{d}{r}}} + \left(\frac{\kappa}{n\alpha_n^{\frac{d}{r}}}\right)^{\frac{q+1}{q+2-d/r}} + \frac{\kappa'}{n\alpha_n^{\frac{d}{r}}} + \frac{\kappa'' x}{n}.$$

The choice of α_n in (11) optimizes the RHS. Integrating with respect to the training set, one leads to the conclusion. ■

6.4 Proof of Theorem 12

To prove Theorem 12, we use a general oracle inequality for aggregation. Let us first recall the general context of aggregation.

Suppose we have $M \geq 2$ differents classifiers $f_1, \ldots, f_M$ with values in $\{-1,1\}$. The method of aggregation consists in building a new classifier $\tilde{f}_n$ from D_n called aggregate which mimics the best among $f_1, \ldots f_M$. Our procedure is using exponential weights of the following form:

$$\omega_j^{(n)} = \frac{\exp\left(\sum_{i=1}^n Y_i f_j(X_i)\right)}{\sum_{k\in\{1\ldots M\}} \exp\left(\sum_{i=1}^n Y_i f_k(X_i)\right)}.$$

Then we define the following aggregate:

$$\tilde{f}_n = \sum_{j=1}^M \omega_j^{(n)} f_j. \tag{21}$$

Under the margin assumption (10), we have this oracle inequality:

Theorem 20 (Lecué, 2005) *Suppose* (10) *holds for some $q \in (0,+\infty)$. Assume we have at least a polynomial number of classifiers to aggregate (i.e., there exist $a \geq 1$, $b > 0$ such that $M \geq an^b$). Then the aggregate defined in* (21) *satisfies, for all integer $n \geq 1$,*

$$\mathbb{E}R(\tilde{f}_n, f^*) \leq (1+2\log^{-1/4} M)\left(2\min_{k\in\{1,\ldots M\}} R(f_k, f^*) + Cn^{-\frac{q+1}{q+2}} \log^{7/4} M\right), \tag{22}$$

where C depends on a, b and the constant c_0 appearing in (10)*.*

Proof (of Theorem 12)

Let $(q_0, s_0) \in K$ and consider $0 < q_{min} < q_{max} < +\infty$ and $0 < s_{min} < s_{max} < +\infty$ such that $K \subset [q_{min}, q_{max}] \times [s_{min}, s_{max}]$. We consider the function

$$\Phi(q,s) = \frac{r(r-s)(q+1)}{s(r(q+2)-d) + (r-s)(q+1)d}$$

defined on $[0,+\infty[\times[0,+\infty[$ with value on $[\frac{1}{2}, \frac{r}{d}]$. We denote by

$$k_0 \in \left\{0, \ldots, \left\lfloor \frac{(2r-d)\Delta}{2d} \right\rfloor - 1\right\}$$

the integer such that

$$\frac{1}{2}+k_0\Delta^{-1}\leq\Phi(q_0,s_0)\leq\frac{1}{2}+(k_0+1)\Delta^{-1}.$$

Since $q\mapsto\Phi(q,s)$ continuously increases on $\mathbb{R}^+$, for n greater than a constant depending on b, r, d and K, there exists $\overline{q}_0\in\left[\frac{q_{min}}{2},q_{max}\right]$ such that $\overline{q}_0\leq q_0$ and

$$\Phi(\overline{q}_0,s_0)=\frac{1}{2}+k_0\Delta^{-1}. \tag{23}$$

Now we can apply Theorem 20 for $\overline{q}_0$. Since $\Delta=n_2^b$, putting $M=\left\lfloor\frac{(2r-d)\Delta}{2d}\right\rfloor$ we have the following oracle inequality:

$$\mathbb{E}_{P^{\otimes n_2}}\left(R(\tilde{f}_n,f^*)|D_{n_1}^1\right)\leq(1+2\log^{-\frac{1}{4}}M)\left(2\min_{\alpha\in\mathcal{G}(n_2)}\left(R(\hat{f}_{n_1}^{\alpha},f^*)\right)+C_1n_2^{-\frac{\overline{q}_0+1}{\overline{q}_0+2}}\log^{7/4}M\right),$$

where C_1 depends on c_0, K and b. Hence we have, integrating with respect to $D_{n_1}^1$,

$$\mathbb{E}\left(R(\tilde{f}_n,f^*)\right)\leq C_2\left(\mathbb{E}R(\hat{f}_{n_1}^{\alpha_{k_0}},f^*)+n_2^{-\frac{\overline{q}_0+1}{\overline{q}_0+2}}\log^{7/4}n_2\right),$$

where $\alpha_{k_0}=m^{-\phi_{k_0}}=n_2^{-\Phi(\overline{q}_0,s_0)}$ with (23) and C_2 depends on K,b,r,d and c_0. Therefore we can apply Theorem 7 to the classifier $\hat{f}_{n_1}^{\alpha_k}$:

$$\mathbb{E}_{P^{\otimes n_1}}R(\hat{f}_{n_1}^{\alpha_{k_0}},f^*)\leq Cn_1^{-\frac{s_0}{r-s_0}\Phi(\overline{q}_0,s_0)},$$

where C depends on r,d and K. Remark that C does not depend on $\overline{q}_0$ and s_0 since $(\overline{q}_0,s_0)\in\left[\frac{q_{min}}{2},q_{max}\right]\times[s_{min},s_{max}]$. Moreover C is uniformly bounded over (q,s) belonging to a compact in Theorem 7.

Finally suppose P satisfies (10) for q_0. Hence we obtain:

$$\begin{aligned}\mathbb{E}\left(R(\tilde{f}_n,f^*)\right)&\leq C_3\left(n_1^{-\frac{s_0}{r-s_0}\Phi(\overline{q}_0,s_0)}+n_2^{-\frac{\overline{q}_0+1}{\overline{q}_0+2}}\log^{\frac{7}{4}}n_2\right)\\&\leq C_3\left(n^{-\frac{s_0}{r-s_0}\Phi(\overline{q}_0,s_0)}+n^{-\frac{\overline{q}_0+1}{\overline{q}_0+2}}\log^{\frac{11}{4}}n\right)\\&\leq C_3'n^{-\frac{s_0}{r-s_0}\Phi(\overline{q}_0,s_0)}.\end{aligned}$$

for $C_3:=C_3(K,b,c_0,r,C_0,d)$. We have $n\geq n_2\geq\frac{an}{\log n}$ and $n_1\geq n(\frac{2}{3}-\frac{a}{\log 3})$. Then for n greater than a constant depending on β_{min}, a, and b, there exists $C_3':=C_3'(K,b,c_0,r,C_0,d)$ such that

The construction of $\overline{q}_0$ and restrictions on r entail $\frac{s_0}{r-s_0}|\Phi(\overline{q}_0,s_0)-\Phi(q_0,s_0)|\leq\Delta^{-1}=n_2^{-b}$. We lead to the conclusion since the sequence $(n^{n_2^{-b}})_{n\in\mathbb{N}}$ is convergent. ■

Acknowledgments

I would like to acknowledge my advisor, Laurent Cavalier, for giving me many ideas and advices for this work. I'm also grateful to Liva Ralaivola for the experimental part of this work and to the anonymous referees for interesting remarks.

References

R.A. Adams. *Sobolev Spaces*. Academic Press, 1975.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54 (2):317–331, 1997.

P.L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44 (2):525–536, 1998.

P.L. Bartlett and S. Mendelson. Empirical minimization. *Probability Theory and Related Fields*, 135 (3):311–334, 2006.

P.L. Bartlett, M.I. Jordan, and J.D. McAuliffe. Convexity, classification, and risk bounds. *J. Amer. Statist. Assoc.*, 101 (473):138–156, 2006.

C. Bennett and R. Sharpley. *Interpolation of Operators*. Academic Press, 1988.

G. Blanchard, O. Bousquet, and P. Massart. Statistical performance of support vector machines. to appear Annals of Statistics, 2006.

B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152, 1992.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: a survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

D.R. Chen, Q. Wu, Y. Ying, and D.X. Zhou. Support vector machine soft margin classifiers: error analysis. *Journal of Machine Learning Research*, 5:1143–1175, 2004.

N. Cristianini and H. Shawe-Taylor. *Introduction to Support Vector Machines, and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

L. Devroye. Necessary and sufficient conditions for the pointwise convergence of nearest neighbor regression function estimates. *Z. Wahrsch. Vew. Gebiete*, 61 (4):467–481, 1982.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121 (2): 256–285, 1995.

A. Karatzoglou, A. Smola, and K. Hornik. An S4 package for kernel methods in R. Reference manual, 2007.

G. Lecué. Simultaneous adaptation to the margin and to complexity in classification. *The Annals of Statistics*, 35 (4):1698–1721, 2007a.

G. Lecué. Optimal rates of aggregation in classification under low noise assumption. *Bernoulli*, 13 (4):1000–1022, 2007b.

P. Malliavin. *Analyse de Fourier-Analyses spectrales*. Ecole Polytechnique, 1974.

E. Mammen and A.B. Tsybakov. Smooth discrimination analysis. *The Annals of Statistics*, 27 (6): 1808–1829, 1999.

P. Massart and E. Nédélec. Risk bounds for statistical learning. *The Annals of Statistics*, 34 (5): 2326–2366, 2006.

M. Matache and V. Matache. Hilbert spaces induced by Toeplitz covariance kernels. *Lecture notes in Control and Information Sciences*, 280:319–334, 2002.

A. Nemirovski. *Topics in Nonparametric Statistics*. Ecole d'été de Saint-Flour XXVIII, Springer, N.Y., 1998.

D. Rätsch, T. Onoda, and K.R. Müller. Soft margin for adaboost. Esprit Working Group in Neural and Computational Learning II, 1998.

S. Smale and D.X. Zhou. Estimating the approximation error in learning theory. *Analysis and Applications*, 1 (1):17–41, 2003.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

I. Steinwart. Consistency of support vector machines and other regularized kernel classifiers. *IEEE Transactions on Information Theory*, 51 (1):128–142, 2005.

I. Steinwart and C. Scovel. Fast rates for support vector machines. In *Proc. 18th Annu. Conference on Comput. Learning Theory*, volume 3559, pages 279–294, 2005.

I. Steinwart and C. Scovel. Fast rates for support vector machines using Gaussian kernels. *The Annals of Statistics*, 35 (2):575–607, 2007.

I. Steinwart, D. Hush, and C. Scovel. An oracle inequality for clipped regularized risk minimizers. *Neural Information Processing Systems*, 19:1321–1328, 2007.

H. Triebel. *Theory of Functions Spaces II*. Birkhauser, 1992.

H. Triebel. *Interpolation Theory, Function Spaces, Differential Operators*. North-Holland Publishing Company, 1978.

A.B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *The Annals of Statistics*, 32 (1):135–166, 2004.

V.N. Vapnik and A.Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16 (2):264–280, 1971.

V.N. Vapnik and A.Ya. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.

R.C. Williamson, A.J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transactions on Information Theory*, 47 (6):2516–2532, 2001.

Q. Wu and D.X. Zhou. Analysis of support vector machine classification. *J. Comput. Anal. Appl.*, 8 (2):99–119, 2006.

Q. Wu, Y. Ying, and D.X. Zhou. Multi-kernel regularized classifiers. *Journal of Complexity*, 23 (1): 108–134, 2007.

Y. Yang. Mixing strategies for density estimation. *The Annals of Statistics*, 28 (1):75–87, 2000.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32 (1):56–85, 2004.

Journal of Machine Learning Research 9 (2008) 1583-1614 Submitted 9/07; Revised 1/08; Published 7/08

Dynamic Hierarchical Markov Random Fields for Integrated Web Data Extraction

Jun Zhu JUN-ZHU@MAILS.TSINGHUA.EDU.CN
Department of Computer Science and Technology
Tsinghua University
Beijing, 100084, China

Zaiqing Nie ZNIE@MICROSOFT.COM
Web Search and Mining Group
Microsoft Research Asia
Beijing, 100080, China

Bo Zhang DCSZB@TSINGHUA.EDU.CN
Department of Computer Science and Technology
Tsinghua University
Beijing, 100084, China

Ji-Rong Wen JRWEN@MICROSOFT.COM
Web Search and Mining Group
Microsoft Research Asia
Beijing, 100080, China

Editor: John Lafferty

Abstract

Existing template-independent web data extraction approaches adopt highly ineffective decoupled strategies—attempting to do data record detection and attribute labeling in two separate phases. In this paper, we propose an integrated web data extraction paradigm with hierarchical models. The proposed model is called Dynamic Hierarchical Markov Random Fields (DHMRFs). DHMRFs take structural uncertainty into consideration and define a joint distribution of both model structure and class labels. The joint distribution is an exponential family distribution. As a conditional model, DHMRFs relax the independence assumption as made in directed models. Since exact inference is intractable, a variational method is developed to learn the model's parameters and to find the MAP model structure and label assignments. We apply DHMRFs to a real-world web data extraction task. Experimental results show that: (1) integrated web data extraction models can achieve significant improvements on both record detection and attribute labeling compared to decoupled models; (2) in diverse web data extraction DHMRFs can potentially address the blocky artifact issue which is suffered by fixed-structured hierarchical models.

Keywords: conditional random fields, dynamic hierarchical Markov random fields, integrated web data extraction, statistical hierarchical modeling, blocky artifact issue

1. Introduction

The World Wide Web is a vast and rapidly growing repository of information. There are various kinds of objects, such as products, people, and conferences, embedded in webpages. Extracting object information is key to object-level search engines like *Libra* (http://libra.msra.cn/) and

Rexa (http://rexa.info). Recent work has shown that *template-independent* approaches to extracting meta-data for the same type of real-world objects are feasible and promising. However, existing approaches use highly ineffective decoupled strategies—attempting to do data record detection and attribute labeling in two separate phases. This paper is to first propose an integrated web data extraction paradigm with hierarchical Markov Random Fields, and then address the blocky artifact issue (Irving et al., 1997) with Dynamic Hierarchical Markov Random Fields.

A Motivating Example: we begin by illustrating the problem with an example, drawn from an actual application of product information extraction under our Windows Live Product Search project (http://products.live.com). The goal is to extract meta-data about real-world products from every product page on the Web. Specifically, for crawled webpages, we first use a classifier to select product pages and then extract the *Name*, *Image*, *Price*, and *Description* of each product from the identified product pages. Our statistical study on 51K randomly crawled webpages shows that about 12.6 percent are product pages. That is, there are about 1 billion product pages within a search index containing 9 billion crawled webpages. If only half of them are correctly extracted, we will have a huge collection of meta-data about real-world products that could be used for further knowledge discovery and data management tasks, such as comparison shopping and user intention detection.

However, how to extract product information from webpages generated by many (maybe tens of thousands of) different templates is non-trivial. One possible solution is that we first distinguish webpages generated by different templates, and then build an extractor for each template; this type of solution is *template-dependent*. Template-dependent methods are impractical for two reasons. First, accurately identifying webpages for each template is a far from trivial task because even webpages from the same website may be generated by dozens of templates. Second, even if we can distinguish webpages, the learning and maintenance of so many different extractors for different templates will require substantial efforts.

Fortunately, recent work (Lerman et al., 2004; Zhai and Liu, 2005; Zhu et al., 2005) has shown the feasibility and promise of *template-independent* meta-data extraction for the same type of objects. We can simply combine the existing techniques to build a template-independent extractor for product pages. Specifically, two types of webpages—*list pages* and *detail pages*[1]—are needed to be treated by existing extraction methods. List pages are webpages containing several structured data records, and detail pages are webpages only containing detailed information about a single object. Figure 1 illustrates these two types of pages. For list pages, we can first use the methods by Zhai and Liu (2005) Lerman et al. (2004) to detect data records and then use the model by Zhu et al. (2005) to label the data elements within the detected records. Similarly, for detail pages we can first use the methods by Song et al. (2004) to identify a main data block of a detail page, and then use the same model from Zhu et al. (2005) to do attribute labeling for the elements in the main block.

However, it is highly ineffective to use decoupled strategies—attempting to do data record detection and attribute labeling in two separate phases. The reasons for this are:

Error Propagation: as record detection and attribute labeling are two separate phases, the errors in record detection will be propagated to attribute labeling. Thus, the overall performance is limited and upper-bounded by that of record detection.

Lack of Semantics in Record Detection: human readers always take into account semantics of the text to understand webpages. For instance, in Figure 1(a), when claiming a block is a data record, we use the evidence that it contains a product's name, image, price, and description. Thus,

1. Our empirical study shows that about 0.35 of product pages are list pages and the rest are detail pages.

(a) A list page with two data records. The first record contains 7 elements and the second contains 8 elements.

(b) A detail page contains one product item.

Figure 1: A sample list page and a detail page.

more effective record detection algorithms should take into account the semantic labels of the text, but existing methods (Zhai and Liu, 2005; Lerman et al., 2004) do not consider them.

Lack of Mutual Interactions in Attribute Labeling: data records in the same page are strongly correlated. They always have a similar layout and the elements at the same position of different records always have similar features and semantic labels. For example, in Figure 1(a) the element on the top-left of each record is an image. Existing methods (Zhu et al., 2005) do not achieve these correlations because data records are labeled independently.

First-Order Markov Assumption: for webpages, especially detail pages, long-distance dependencies always exist between different attribute elements. This is because there are always many irrelevant elements or noise elements appearing between the attributes. For example, in Figure 1(b) there are several noise elements, such as "Add to Cart" and "Select Quantity", appearing between the price and description. However, plat models like 2D CRFs (Zhu et al., 2005) cannot incorporate long-distance dependencies because of their first-order Markov assumption.

To address the above problems, the first part of this paper is to propose an integrated web data extraction paradigm. Specifically, we take a vision-tree representation of webpages and define both record detection and attribute labeling as assigning semantic labels to the nodes on the trees. Then, we can define the integrated web data extraction that performs record detection and attribute labeling simultaneously. Based on the tree representation, we define a simple integrated web data extraction model—Hierarchical Conditional Random Fields (HCRFs), whose structures are determined by vision-trees.

However, for HCRFs, their structures may not be the most appropriate for web data extraction. This is because when constructing the vision-tree of each webpage, it is unaware of semantic labels. Thus, they cannot resolve all ambiguities. This will lead to those cases in which some closely related nodes may be separated significantly and only connected through a remote ancestor node on the tree. Due to the model's local Markov assumption, it will lose some useful dependencies and result in low accuracy. An extreme case is that the attributes of different objects are intertwined. Figure 2 shows an example where the two neighboring records on the webpage have their attributes intertwined on the corresponding tree. In this case, fixed-structured hierarchical models are incapable of reorganizing them correctly. This problem has been generally known as blocky artifact issue in image processing (Irving et al., 1997).

Thus, effective web data extraction models should have the capability to adapt their structures during the inference process. The second part of this paper is to generalize Hierarchical Conditional Random Fields to incorporate structural uncertainty. The general model is called Dynamic Hierarchical Markov Random Fields (DHMRFs). DHMRFs consist of two parts—structure model and class label model. Both parts are jointly defined as an exponential family distribution. Compared to the directed Dynamic Trees (Williams and Adams, 1999) which have been proposed in image processing to address the blocky artifact issue, our model representation is compact and parameter sharing is easy. This is because conditional probability tables (CPTs) are used in Dynamic Trees to represent transition from parent nodes to child nodes. If different CPTs are used for different nodes, it will easily lead to over-parameterization. Thus, layer-wise CPT sharing is always adopted. But in the scenario of web data, sharing CPTs can be difficult because the hierarchical structures are not as regular as the dyadic or quad trees in image processing. Here, different pages can have quite different depths, and nodes from different pages at the same depth can have very diverse semantics. In contrast, DHMRFs define probability distributions via a set of feature functions and weights. These feature functions depend much more on observations and their labels than on the depths of the nodes. Thus, the undirected model is more suitable for diverse web data extraction. Furthermore, as a conditional model (Lafferty et al., 2001), DHMRFs relax the conditional independence assumption among observations as made in directed models. Finally, instead of trees in which only parent-child dependencies are assumed, DHMRFs consider the triple-wise interactions among neighboring sibling variables and their parent. These triple-wise dependencies provide more flexibility in encoding useful features.

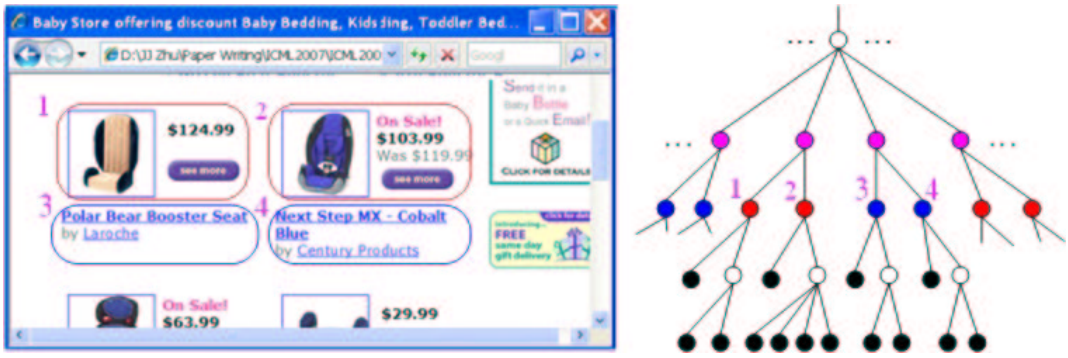


Figure 2: An intertwined example webpage. Blocks 1 and 3 present information of one product and blocks 2 and 4 present information of another product. But on the right tree, the information is not correctly grouped.

In undirected dynamic models, parameter estimation is generally intractable, especially when there are hidden variables—both structures and inner variables are hidden in our study. We develop a variational algorithm within the paradigm of contrastive divergence mean field learning (Welling and Hinton, 2001) to do parameter estimation and to find the MAP assignment of labels and the most likely model structures. The performance of our models is demonstrated on a web data extraction task—production information extraction. The results show that: (1) integrated web data extraction models can significantly improve the performance of both record detection and attribute labeling compared to decoupled methods; (2) Dynamic Hierarchical Markov Random Fields can (partially) avoid the blocky artifact issue and achieve high extraction accuracy without tedious manual labeling of inner nodes, which is required in the learning of the fixed-structured models; (3) integrated extraction models can generalize well to unseen templates. Note that the model is general and could be applied to other fields. We leave further examinations as future work.

The rest of the paper is organized as follows. In the next section, we discuss some background knowledge on which this work is based. Section 3 presents an integrated web data extraction paradigm and fixed-structured Hierarchical Conditional Random Fields. Section 4 describes Dynamic Hierarchical Markov Random Fields, including an approximate inference algorithm. Section 5 describes implementation details and experimental setup on the task of product information extraction. Section 6 and 7 presents evaluation results. Section 8 brings this paper to a conclusion and some future research directions are discussed. Finally, we give our acknowledgements.

2. Preliminary Background Knowledge

The background knowledge, on which the following work is based, is from web data extraction and statistical hierarchical modeling. We introduce these two fields in turn.

2.1 Web Data Extraction

Web data extraction is an information extraction (IE) task that identifies information of interest from webpages. The difference of web data extraction from traditional IE is that various types of

structural dependencies between HTML elements exist. For example, the HTML tag tree is itself hierarchical and each webpage is displayed as a two-dimensional image to readers. Leveraging the two-dimensional spatial information to extract web data has been studied (Zhu et al., 2005; Gatterbauer et al., 2007). This paper is to explore both hierarchical and two-dimensional spatial information for more effective web data extraction.

Wrapper learning approaches (Muslea et al., 2001; Kushmerick, 2000) are template-dependent. They take in some manually labeled webpages and learn some extraction rules (or wrappers). Since the learned wrappers can only be used to extract data from similar pages, maintaining the wrappers as web sites change will require substantial efforts. Furthermore, in wrapper learning users must provide explicit information about each template. So it will be expensive to train a system that extracts data from many web sites. The methods by Embley et al. (1999), Buttler et al. (2001), Chang and Lui (2001), Crescenzi et al. (2001) and Arasu and Garcia-Molina (2003) are also template-dependent, but they do not need labeled training data. They produce wrappers from a collection of similar webpages.

The methods by Zhai and Liu (2005), Lerman et al. (2004) and Gatterbauer et al. (2007) are template-independent. In work by Lerman et al. (2004), data on list pages are segmented using the information from their detail pages. The need of detail pages is a limitation because automatically identifying links that point to detail pages is non-trivial and there are also many pages that do not have detail pages behind them. Zhai and Liu (2005) proposed to detect data records using string matching and also some visual features to achieve better performance, but no semantics are considered. Like the work by Zhu et al. (2005), a general 2D visual model was proposed by Gatterbauer et al. (2007) to extract web tables. The data extracted by the methods of Zhai and Liu (2005), Lerman et al. (2004) and Gatterbauer et al. (2007) have no semantic labels. Our work (Zhu et al., 2005) is complementary to this and assigns semantic labels to the extracted data.

2.2 Statistical Hierarchical Modeling

Multi-scale or hierarchical statistical modeling has shown great promise in image labeling (Kato et al., 1993; Li et al., 2000; He et al., 2004; Kumar and Hebert, 2005) and human activity recognition (Liao et al., 2005). Based on whether data are observed at multiple scales, two scenarios exist in which hierarchical modeling is appropriate. First, data are observed at different spatial scales and a model is used to integrate information from the different scales. Second, data are observed only at the finest scale and a model is used to induce a particular process at that scale. The introduced intermediate processes or variables can incorporate more complex dependencies to help the target labeling. Another merit of hierarchical models is that they admit more efficient inference algorithms compared to flat models (Willsky, 2002).

Traditional hierarchical models always assume that model structures are fixed or can be constructed via some deterministic methods, such as sub-sampling of images (Li et al., 2000) and the minimum spanning tree algorithm (Quattoni et al., 2004) with a proper definition of distance. However, in many applications this assumption may not hold. For example, fixed models in image processing often lead to the blocky artifact issue, and the similar problem arises in web data extraction due to the diversity of web data. To address this problem some enhanced models have been proposed, such as the overlapping tree approach (Irving et al., 1997). Superior performance is achieved with the improvement of the descriptive component of the model. However, ultimate solutions should deal with the source of the blockiness—fixed model structures. Based on this intu-

ition, Dynamic Trees (Williams and Adams, 1999) have been proposed, which also consist of two parts—model of structures and model of class labels. However, the difference between DHMRFs and Dynamic Trees is that DHMRFs are defined as exponential family distributions and thus admit several advantages as discussed in the introduction.

Incorporating evidence at various scales was examined in a generative manner by Todorovic and Nechyba (2005). But our model is discriminative and it can relax the independence assumption among evidence as made in generative models. This is the key idea underlying Conditional Random Fields (Lafferty et al., 2001), which have shown great promise in information extraction (Culotta et al., 2006; Zhu et al., 2005). Modeling structural uncertainty has also been studied in relational learning (Getoor et al., 2001). Here, we focus on modeling the structural uncertainty within independently and identically distributed samples.

Finally, the work has partially appeared in the conference papers Zhu et al. (2006) and Zhu et al. (2007b).

3. Integrated Web Data Extraction

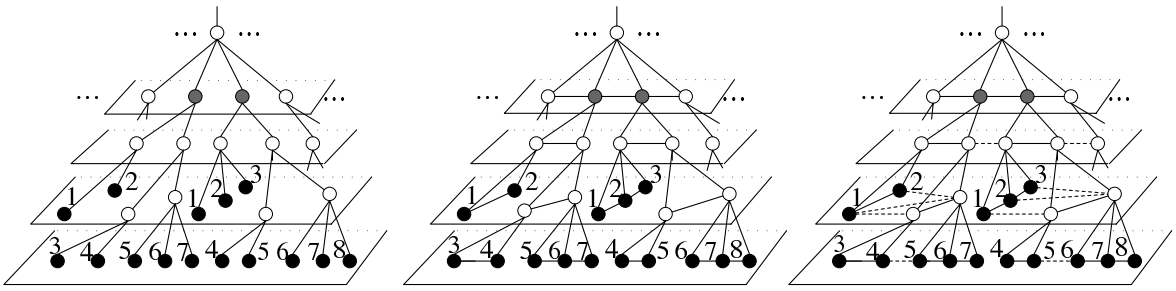
In this section, we formally define the integrated web data extraction and also propose Hierarchical Conditional Random Fields (HCRFs) to perform that task.

3.1 Vision-Tree Representation

For web data extraction, the first thing is to find a good representation format for webpages. Good representation can make the extraction task easier and improve extraction accuracy. In most previous work, tag-tree, which is a natural representation of the tag structure, is commonly used to represent a webpage. However, as Cai et al. (2004) pointed out, tag-trees tend to reveal presentation structure rather than content structure, and are often not accurate enough to discriminate different semantic portions in a webpage. Moreover, since authors use different styles to compose webpages, tag-trees are often complex and diverse. To overcome these difficulties, Cai et al. (2004) proposed a vision-based page segmentation (VIPS) approach. VIPS makes use of page layout features such as font, color, and size to construct a vision-tree for a page. It first extracts all suitable nodes from the tag-tree and then finds separators between these nodes. Here, separators denote horizontal or vertical lines in a webpage that visually do not cross any node. Based on these separators, the vision-tree of the webpage is constructed. Each node on this tree represents a data region in the webpage, which is called a block. The root block represents the whole page. Each inner block is the aggregation of all its child blocks. All leaf blocks are atomic units (i.e., elements) and form a flat segmentation of the webpage. Since vision-tree can effectively keep related content together while separating semantically different blocks from one another, we use it as our data representation format. Figure 3(a) is a vision-tree for the page in Figure 1(a), where empty circles denote inner blocks and filled circles denote leaf blocks (elements). For simplicity, we only show a sub-tree which contains the two data records in Figure 1(a). A detailed example was provided by Cai et al. (2004).

3.2 Record Detection and Attribute Labeling

Based on the definition of vision-tree, we now formally define the concepts of record detection and attribute labeling.



(a) (b) (c)

Figure 3: (a) Partial vision-tree of the webpage in Figure 1(a); (b) An HCRF model with linear-chain neighborhood between sibling nodes; (c) Another HCRF model with 2D neighborhood between sibling nodes and between nodes that share a grand-parent. Here, filled circles denote leaf blocks (elements) and the variables associated with them. Each filled circle corresponds to an element in the page in Figure 1(a) with the same number. Empty circles represent inner nodes and inner variables. The two gray nodes in each chart denote the roots of the sub-trees that correspond to the two data records in Figure 1(a).

Definition 3.1 (Record detection): Given a vision-tree, record detection is the task of locating the root of a minimal subtree that contains the content of a record. For a list page containing multiple records, all the records need to be identified.

For instance, for the vision-tree in Figure 3(a), the two blocks in gray are detected as data records. Note that as shown in Figure 2, given a particular vision-tree, we are not guaranteed to find the root nodes that correspond to data records. This is the very problem to be addressed by Dynamic Hierarchical Markov Random Fields.

Definition 3.2 (Attribute labeling): For each identified record, attribute labeling is the task of assigning attribute labels to the leaf blocks (elements) within the record.

We can build a complete model to extract both records and attributes by sequentially combining existing record detection and attribute labeling algorithms. However, as we have stated, this decoupled strategy is highly ineffective. Therefore, we propose an integrated approach that conducts simultaneous record extraction and attribute labeling.

3.3 Integrated Web Data Extraction

Based on the above definitions, both record detection and attribute labeling are the task of assigning labels to blocks of the vision-tree for a webpage. Therefore, we can define one probabilistic model to deal with both tasks. Formally, we define the integrated web data extraction as:

Definition 3.3 (Integrated Web Data Extraction): Given a vision-tree of a page, let $\mathrm{x} = \{x_0, x_1, \ldots, x_N\}$ be the features of all the blocks and each component x_i is a feature vector of one block, and let $\mathrm{y} = \{y_0, y_1, \ldots, y_N\}$ be one possible label assignment of the corresponding blocks. The goal of web data extraction is to find a label assignment $\mathrm{y}^\star$ that has the maximum posterior probability $\mathrm{y}^\star = \arg\max_{\mathrm{y}} p(\mathrm{y}|\mathrm{x})$, and extract data from this assignment.

3.4 Hierarchical Conditional Random Fields

In this section, we first introduce some basics of Conditional Random Fields and then propose Hierarchical Conditional Random Fields for integrated web data extraction.

3.4.1 CONDITIONAL RANDOM FIELDS

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are Markov Random Fields that are globally conditioned on observations. Let $G = (V,E)$ be an undirected model over a set of random variables X and Y. X are variables over the observations to be labeled and Y are variables over the corresponding labels. The random variables Y could have a non-trivial structure, such as a linear-chain (Lafferty et al., 2001) and a 2D grid (Zhu et al., 2005). Each component Y_i has a label space or the set of possible labels $\mathcal{Y}_i$. The conditional distribution of the labels y (an instance of Y) given the observations x (an instance of X) has the form

$$p(\mathrm{y}|\mathrm{x}) = \frac{1}{Z(x)} \prod_{c \in \mathcal{C}} \phi(\mathrm{y}|_c, \mathrm{x}),$$

where $\mathcal{C}$ is the set of cliques in G; $\mathrm{y}|_c$ are the components of y associated with the clique c; ϕ is a potential function taking non-negative real values; $Z(\mathrm{x}) = \sum_{\mathrm{y}} \prod_{c \in \mathcal{C}} \phi(\mathrm{y}|_c, \mathrm{x})$ is the normalization factor or partition function in physics. The potential functions are usually expressed in terms of feature functions $f_k(\mathrm{y}|_c, \mathrm{x})$ and their weights λ_k:

$$\phi(\mathrm{y}|_c, \mathrm{x}) = \exp\Big\{ \sum_k \lambda_k f_k(\mathrm{y}|_c, \mathrm{x}) \Big\}.$$

Although functions f_k can take any real value, here we assume they are boolean and take either true or false.

3.4.2 HIERARCHICAL CONDITIONAL RANDOM FIELDS

Based on the vision-tree representation of the data, a Hierarchical Conditional Random Field (HCRF) model can be easily constructed. For the page in Figure 1(a) and its corresponding tree in Figure 3(a), an HCRF model is shown in Figure 3(b), where we also use empty circles to denote inner nodes and use filled circles to denote leaf nodes. For simplicity, only part of the model graph is presented. Each node on the graph is associated with a random variable Y_i. We will use nodes and variables exchangeably when there is no ambiguity. The observations that are globally conditioned on are omitted from this graph for simplicity. To make the model simple, we assume that the inner-layer interactions among sibling variables are sequential, that is, sibling variables are put into a sequence and only the relationships between neighboring variables are considered. Here, we use the position information and sequentialize the elements from left to right, top to bottom. For easy explanation and implementation, we assume that every inner node contains at least two children. Otherwise, we replace the parent with its single child. This assumption has no affect on the performance because the parent is identical to its child in this case.

The cliques of the graph in Figure 3(b) are its vertices, edges, and triangles. Let L be the number of layers indexed from 0 to $L-1$ starting from the root, and each layer $d (0 \le d < L)$ has N_d nodes. Let s_{il} be an indicator variable to denote the connectivity between node i and node l, where l is at the direct above layer of i. Let n_{ij} be an indicator variable to denote whether node i and node j are

adjacent to each other at the same layer. Then, $T = \cup_{d=1}^{L-1}\{(i,j,l) : 0 \le i,j < N_d, 0 \le l < N_{d-1}, n_{ij} = 1, s_{il} = 1, and\ s_{jl} = 1\}$ is the set of triangles in the graph G. Thus, $\mathcal{C} = V \cup E \cup T$ and the conditional probability is

$$p(\mathrm{y}|\mathrm{x}) = \frac{1}{Z(x)} \exp\Big\{ \sum_{v\in V}\sum_{k} \mu_k g_k(\mathrm{y}|_v, \mathrm{x}) + \sum_{e\in E}\sum_{k} \lambda_k f_k(\mathrm{y}|_e, \mathrm{x}) + \sum_{t\in T}\sum_{k} \gamma_k h_k(\mathrm{y}|_t, \mathrm{x}) \Big\}.$$

Note that we use the same notation Z to denote the normalization factor for both CRFs and HCRFs, although they are different. We will follow this notation when there is no ambiguity in the rest of the paper.

Figure 3(c) presents another slightly more complicated HCRF model. In this model, we consider the two-dimensional inner-layer dependency relationships between sibling nodes. Moreover, we also consider the two-dimensional interactions between nodes that share a common grant-parent on the tree. In Figure 3(c), dotted edges are introduced to encode additional dependencies compared to the model in Figure 3(b). The conditional probability $p(\mathrm{y}|\mathrm{x})$ is the same as that of the previous model but with the dotted edges included in E.

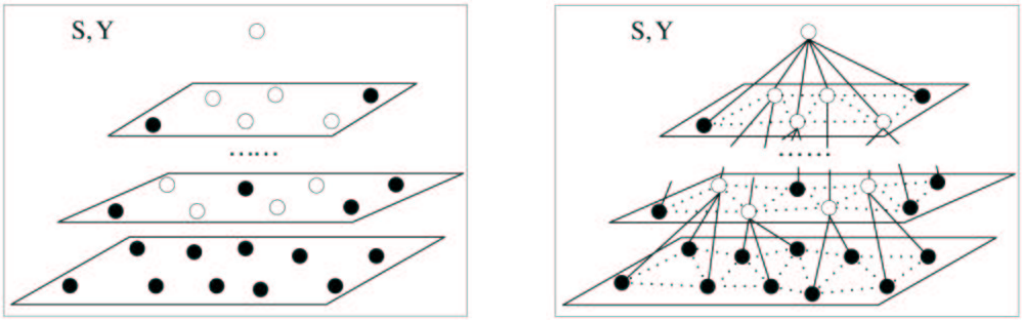
For the model in Figure 3(b), the graph is a chordal graph and its inference can be exactly and efficiently done with the junction tree algorithm (Cowell et al., 1999). In fact, the complexity of the junction tree algorithm is linear in terms of the number of maximum cliques (or triangles), which can be shown to be equivalent to the number of leaf nodes (or elements). For the model in Figure 3(c), however, no exact inference algorithm exists; we have to turn to approximate algorithms. Since the backbone (without dotted edges) of the model graph is the same as the previous model, whose inference can be exactly done, piecewise learning (Sutton and McCallum, 2005) should be a good method. The basic idea of piecewise learning is to partition the graph into a set of disjointed small pieces. For each piece, exact inference can be efficiently done. Then, a lower bound of the log-likelihood function can be derived as the combination of the local log-likelihoods on different pieces. To use piecewise learning, here, we take the backbone as one piece and take each additional edge (a dotted edge) as one piece. The method by Wainwright et al. (2002) could be another excellent approximate algorithm in our model. Unlike piecewise learning whose parameter estimation is still a maximization problem, the parameter estimation by Wainwright et al. (2002) becomes a constrained saddle point problem.

4. Dynamic Hierarchical Markov Random Fields

In this section, we present the detailed description of Dynamic Hierarchical Markov Random Fields. An approximate inference algorithm is developed to perform parameter estimation and to find the maximum a posterior model structure and label assignment.

4.1 Model Description

Suppose we are given a set of N vertices, and each vertex is associated with a set of observations. Also suppose the vertices are arranged in a layered manner. Then, hierarchical statistical modeling is a task to construct an appropriate hierarchical model structure and carry out inference about the labels of given observations. Determining the number of layers and the number of nodes at each layer is problem specific. We will give an example of web data extraction in the experiment section. Let S be random variables over hierarchical structures, X be variables over the observations to be labeled, and Y be variables over the corresponding labels. Each component Y_i is assumed to take



(a) (b)

Figure 4: (a) The initial setting of DHMRFs with a set of nodes that are arranged in multi-layers. Filled circles denote leaf nodes or elements and empty circles denote inner blocks of a webpage; (b) An instance of DHMRFs denoted by S and Y. Vertical edges are selected by posterior probabilities $p(\mathrm{s}|\mathrm{x})$. Dotted lines represent the 2D neighborhood system between nodes at the same layer.

values from a finite discrete label space $\mathcal{Y}_i$. Here, capitalized characters denote random variables and corresponding lower cases are their instances or configurations, for example, y is a label assignment and $y_i \in \mathcal{Y}_i$ is one component label. A state of the system is the pairing of a model structure and a label assignment, that is, (s,y). Given observations x, Dynamic Hierarchical Markov Random Fields (DHMRFs) define a conditional probability distribution $p(\mathrm{s},\mathrm{y}|\mathrm{x})$ of structure s and label assignment y. An example is shown in Figure 4, where the left graph is the initial setting of DHMRFs with a set of nodes that are arranged in multi-layers and the right is an instance of the dynamic model. Let the energy of the system being at the state (s,y) be $E(\mathrm{s},\mathrm{y},\mathrm{x})$, then the probability of the system being at this state is

$$p(\mathrm{s},\mathrm{y}|\mathrm{x}) = \frac{1}{Z(\mathrm{x})}\exp\{-E(\mathrm{s},\mathrm{y},\mathrm{x})\}.$$

This is a Boltzmann distribution with the temperature $T = 1$, and our model is one type of exponential random graph model (Robins et al., 2006). Since the system consists of two parts, the energy is also from two parts. We explain them as follows:

Structure Model: Let s_{il} be an indicator variable to denote the connectivity between node i and another node l, which is at the direct above level. s_{il} equals to 1 if node i connects to node l; otherwise it is 0. Here, leaf nodes can be at any level except the root node that is taken as a default node for an entire page. For leaf nodes, no child is allowed. We call the parent-child connection *vertical connection*. To retain the computational advantage of tree-structured models, each node is allowed to have only one parent in a particular structure s. We will use s_v to denote the set of vertical connections. With the aforementioned definitions of L and N_d, we get $\mathrm{s}_v = \cup_{d=1}^{L-1}\{s_{il} : 0 \leq i < N_d \text{ and } 0 \leq l < N_{d-1}\}$.

To consider the dependencies between the nodes at the same layer, *horizontal connection*s (i.e., connections between nodes at the same level) are incorporated in s. Let n_{ij} be an indicator variable to denote whether node i and node j are adjacent to each other. Similarly, n_{ij} equals to 1 if node

i connects to node j; otherwise, it is 0. Let's denote the set of horizontal connections by s_h, then $\mathrm{s}_h = \cup_{d=0}^{L-1}\{n_{ij} : 0 \le i, j < N_d \text{ and } i \neq j\}$. Here, we assume that the variables n_{ij} are independent of s_{il} and can be determined using some spatial ordering method. This assumption holds in applications such as web data extraction and image processing. As position information is encoded in each node, deterministic spatial ordering can decide the neighborhood system among a set of nodes. In theory, the horizontal neighborhood system can be arbitrary. We consider the 2D cases (Zhu et al., 2005), that is, each node is horizontally connected to all the nearest surrounding nodes in a 2D plane.

With the structure model, the first part of the energy when the system is at the state (s,y) is

$$E_1(\mathrm{s},\mathrm{y},\mathrm{x}) = \sum_k \mu_k \sum_{ijl} s_{il} s_{jl} n_{ij} g_k(i,j,l,\mathrm{x}),$$

where a triple (i,j,l) denotes a particular position in the dynamic model. A position can be a time interval in time series or a region of space in random fields. Here, i and j are two nodes at the same layer and l is a node at the direct above layer. g_k are feature functions defined on the three nodes at position (i,j,l), and μ_k are their weights.

Class Label Model: A sample s from the structure model defines a Hierarchical Conditional Random Field, which has been defined in Section 3.4.2. Let α_i^y be an indicator variable to denote the variable Y_i taking the class label y. Then, the second part of the energy when the system is at the state (s,y) is

$$E_2(\mathrm{s},\mathrm{y},\mathrm{x}) = \sum_k \lambda_k \sum_{ijl} s_{il} s_{jl} n_{ij} \sum_{y_i,y_j,y_l} \alpha_i^{y_i} \alpha_j^{y_j} \alpha_l^{y_l} f_k(y_i,y_j,y_l,\mathrm{x}),$$

where f_k are feature functions defined on the labels y_i, y_j, and y_l at position (i,j,l), and λ_k are their weights.

Although conditional models take observations as global conditions, when defining feature functions they need to know the "focused observations" at a particular position. For example, in linear-chain CRFs (Lafferty et al., 2001) the observation at time t is among the focused observations when defining feature functions related to the label y_t. In general, let t be a position and x_t be the set of focused observations at that position. The mapping function $\zeta : t \to \mathrm{x}_t$ defines the focused observations for each position. In generative models (Todorovic and Nechyba, 2005), the mapping function is defined to determine the observations generated by the states at a particular position. Moreover, an additional constraint $\forall t \neq s, \mathrm{x}_t \cap \mathrm{x}_s = \emptyset$ is also set due to their independence assumption that observations at different positions are conditionally independent given the states at those positions. In conditional models, however, there is no such constraint. The mapping function can be deterministic or stochastic. We assume it to be deterministic in this paper. Now, all feature functions take an additional argument ζ, that is, the feature functions are $g_k(i,j,l,\mathrm{x},\zeta)$ and $f_k(y_i,y_j,y_l,\mathrm{x},\zeta)$.

Taking E_1 and E_2 together, we get the joint distribution of s and y

$$p(\mathrm{s},\mathrm{y}|\mathrm{x}) = \frac{1}{Z(\mathrm{x})} \exp\left\{ \begin{array}{c} \sum_k \mu_k \sum_{ijl} s_{il} s_{jl} n_{ij} g_k(i,j,l,\mathrm{x},\zeta) + \\ \sum_k \lambda_k \sum_{ijl} s_{il} s_{jl} n_{ij} \sum_{y_i,y_j,y_l} \alpha_i^{y_i} \alpha_j^{y_j} \alpha_l^{y_l} f_k(y_i,y_j,y_l,\mathrm{x},\zeta) \end{array} \right\},$$

where $Z(\mathrm{x})$ is the normalization factor or partition function in physics. Note that although names are similar, Dynamic Hierarchical Markov Random Fields are quite different from Dynamic CRFs (Sutton et al., 2004), which are dynamic in terms of time, that is, they have repetitive model structure and parameters over time, and the structure at each time slice is fixed. Here, "Dynamic" means the model's structure is dynamically selected.

4.2 Parameter Estimation and Labeling

Let $\Theta = \{\mu_1, \mu_2, \ldots; \lambda_1, \lambda_2, \ldots\}$ denote the whole set of the model's parameters, and let $D = \{(\mathrm{x}^i, \mathrm{y}_e^i)\}_{i=1}^K$ denote the set of training data, where x^i is a sample and y_e^i are observed labels. We consider the general case with both hidden hierarchical structure s and hidden labels y_h. For example, in web data extraction only the labels of leaf nodes are observable and both the hierarchical structures and the labels of inner nodes are hidden. So the log-likelihood of the data is incomplete

$$L(\Theta) = \sum_{i=1}^{K} \log p(\mathrm{y}_e^i|\mathrm{x}^i) = \sum_{i=1}^{K} \log(\sum_{\mathrm{s},\mathrm{y}_h} p(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e^i|\mathrm{x}^i)).$$

This function does not have a closed-form solution because of the marginalization taking place within the logarithm. In the following, we derive a lower bound of the log-likelihood, or equivalently an upper bound of the negative log-likelihood. Then, contrastive divergence learning (Hinton, 2002) is applied as an approximation.

Let $q(\mathrm{s},\mathrm{y}_h|\mathrm{y}_e,\mathrm{x})$ be an approximation of the true distribution $p(\mathrm{s},\mathrm{y}_h|\mathrm{y}_e,\mathrm{x})$. With a little abuse of notations, we will use $q(\mathrm{s},\mathrm{y}_h)$ to denote $q(\mathrm{s},\mathrm{y}_h|\mathrm{y}_e,\mathrm{x})$. We also ignore the summation operator in the log-likelihood during the following derivations, as there is no essential difference between one sample and a set of independently and identically distributed (IID) samples. The optimal approximation is the distribution that has the minimum Kullback-Leibler divergence between $q(\mathrm{s},\mathrm{y}_h)$ and $p(\mathrm{s},\mathrm{y}_h|\mathrm{y}_e,\mathrm{x})$. The KL divergence is defined as $KL(q||p) = \sum_{\mathrm{s},\mathrm{y}_h} q(\mathrm{s},\mathrm{y}_h) \log \frac{q(\mathrm{s},\mathrm{y}_h)}{p(\mathrm{s},\mathrm{y}_h|\mathrm{y}_e,\mathrm{x})}$.

Take $p(\mathrm{s},\mathrm{y}_h|\mathrm{y}_e,\mathrm{x}) = p(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e|\mathrm{x})/p(\mathrm{y}_e|\mathrm{x})$ into the above equation and use the non-negativity of KL divergence, we get a lower bound of the log-likelihood

$$\log p(\mathrm{y}_e|\mathrm{x}) \geq \sum_{\mathrm{s},\mathrm{y}_h} q(\mathrm{s},\mathrm{y}_h)[\log p(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e|\mathrm{x}) - \log q(\mathrm{s},\mathrm{y}_h)].$$

Equivalently, $\mathcal{L}(\Theta) \triangleq \sum_{\mathrm{s},\mathrm{y}_h} q(\mathrm{s},\mathrm{y}_h)[\log q(\mathrm{s},\mathrm{y}_h) - \log p(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e|\mathrm{x})]$ is an upper bound of the negative log-likelihood $-L(\Theta)$. By analogy with statistical physics, the upper bound, which is actually a KL divergence, can be expressed as the difference of two free energies: $\mathcal{L}(\Theta) = F_0 - F_\infty$, where the first term is the free energy when we use data distribution with observable labels clamped to their values, and the second $F_\infty = -\log Z(\mathrm{x})$ is the free energy when we use model distribution with all variables free.

Now, the problem is to minimize the upper bound. The derivatives of $\mathcal{L}(\Theta)$ with respect to λ_k are

$$\begin{aligned}
\frac{\partial \mathcal{L}(\Theta)}{\partial \lambda_k} &= \frac{\partial}{\partial \lambda_k} \langle -\log p(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e|\mathrm{x}) \rangle_{q(\mathrm{s},\mathrm{y}_h)} \\
&= -\sum_{ijl} \langle s_{il} s_{jl} n_{ij} \rangle_{q(\mathrm{s},\mathrm{y}_h)} \sum_{y_i,y_j,y_l} \langle \alpha_i^{y_i} \alpha_j^{y_j} \alpha_l^{y_l} \rangle_{q(\mathrm{s},\mathrm{y}_h)} f_k(y_i,y_j,y_l,\mathrm{x},\zeta) - \frac{\partial F_\infty}{\partial \lambda_k} \\
&= -\sum_{ijl} n_{ij} \langle s_{il} s_{jl} \rangle_{q(\mathrm{s},\mathrm{y}_h)} \sum_{y_i,y_j,y_l} \langle \alpha_i^{y_i} \alpha_j^{y_j} \alpha_l^{y_l} \rangle_{q(\mathrm{s},\mathrm{y}_h)} f_k(y_i,y_j,y_l,\mathrm{x},\zeta) - \frac{\partial F_\infty}{\partial \lambda_k}, \qquad (1)
\end{aligned}$$

where $\langle . \rangle_p$ is the expectation under the distribution p. The last equality holds because of the assumption that the neighborhood system between sibling nodes is determined independent of their parents.

Similarly, the derivatives of $\mathcal{L}(\Theta)$ with respect to μ_k are

$$\frac{\partial \mathcal{L}(\Theta)}{\partial \mu_k} = -\sum_{ijl} n_{ij} \langle s_{il} s_{jl} \rangle_{q(\mathrm{s},\mathrm{y}_h)} g_k(i,j,l,\mathrm{x},\zeta) - \frac{\partial F_\infty}{\partial \mu_k}. \tag{2}$$

In (1) and (2), the derivatives of the equilibrium free energy F_∞ are intractable in the case of Dynamic Hierarchical Markov Random Fields. However, by viewing the equilibrium distribution as the distribution of a Markov chain at time $t = \infty$ starting with data distribution, Markov chain Monte Carlo (MCMC) method can be used to reconstruct an approximation distribution $q_i(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e)$ within several steps. This is the basic idea of contrastive divergence learning (Hinton, 2002). Now, the upper bound is approximated by

$$\begin{aligned}\mathcal{L}(\Theta) =& F_0 - F_\infty \\ \approx& F_0 - F_i = KL(q_0||p) - KL(q_i||p) \triangleq CF_i^{App},\end{aligned}$$

where $q_0 = q(\mathrm{s},\mathrm{y}_h)$ is optimized with observable labels clamped to their values, and $q_i(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e)$ is optimized with all variables free starting with q_0. As shown by Hinton (2002), CF_i^{App}, known as contrastive divergence, is non-negative. But since $F_i \geq F_\infty$, there is no guarantee that it is still an upper bound. Some analyses of contrastive divergence learning (Yuille, 2004; Carreira-Perpinan and Hinton, 2005) have been carried out. In the sequel, we will set $i = 1$.

Now, the derivatives of CF_1^{App} with respect to the model's parameters are as in (1) and (2) but with the derivatives of F_∞ replaced by

$$-\sum_{ijl} n_{ij} \langle s_{il} s_{jl} \rangle_{q_1} \sum_{y_i y_j y_l} \langle \alpha_i^{y_i} \alpha_j^{y_j} \alpha_l^{y_l} \rangle_{q_1} f_k(y_i, y_j, y_l, \mathrm{x}, \zeta) \text{ and } -\sum_{ijl} n_{ij} \langle s_{il} s_{jl} \rangle_{q_1} g_k(i,j,l,\mathrm{x},\zeta)$$

respectively.

Generally, stochastic sampling is quite time demanding in constructing q_1. In contrast, the deterministic mean field variant (Welling and Hinton, 2001) is more efficient. An extension to the combination of a general deterministic variational approximation and contrastive divergence is studied by Welling and Sutton (2005). The learning procedure consists of two phases—wake phase and sleep phase. Wake phase is to optimize q_0 and sleep phase is to optimize q_1. We address the wake phase first.

Assume q_0 can be factorized as $q_0 = q(\mathrm{s},\mathrm{y}_h) = q(\mathrm{s})q(\mathrm{y}_h)$, and we get

$$KL(q_0||p) = -\langle \log p(\mathrm{s},\mathrm{y}_h,\mathrm{y}_e|\mathrm{x}) \rangle_{q_0} - H(q(\mathrm{s})) - H(q(\mathrm{y}_h)), \tag{3}$$

where $H(p) = -\langle \log p \rangle_p$ is the entropy of distribution p. To efficiently optimize q_0, more assumptions need to be made about the family of distributions of $q(\mathrm{s})$ and $q(\mathrm{y}_h)$. Here, we adopt the naïve mean field approximation. The basic idea underlying mean field theory (Jordan et al., 1999) is to make a distribution a factorized one by introducing additional independence assumptions. This factorized distribution leads to computational tractability. The simplest naïve mean field is to assume that interacted variables are mutually independent and the joint distribution is the product of single variable marginal probabilities.

Let μ_{il} be the probability of node i being connected to node l, and m_i^y be the probability of variable Y_i being at state y. As we assume variables n_{ij} are determined independent of s_{il}, the mean field distributions[2] are

$$q(\mathrm{s}) = \prod_{il} [\mu_{il}]^{s_{il}} \text{ and } q(\mathrm{y}_h) = \prod_{iy} [m_i^y]^{\alpha_i^y}.$$

Substitute the above distributions into (3) and keep $q(\mathrm{y}_h)$ fixed, then we get

$$KL(q_0||p) = -\langle \log p(\mathrm{s}, \mathrm{y}_h, \mathrm{y}_e|\mathrm{x}) \rangle_{q_0} - H(q(\mathrm{s})) + c,$$

where c is a constant. Let the derivative over μ_{il} equal zero, and we get

$$\mu_{il} \propto \exp \left\{ \begin{array}{c} \sum_k \mu_k s_{il} \sum_j \langle s_{jl} \rangle_{q(\mathrm{s})} n_{ij} g_k(i,j,l,\mathrm{x}) + \\ \sum_k \lambda_k s_{il} \sum_j \langle s_{jl} \rangle_{q(\mathrm{s})} n_{ij} \sum_{y_1,y_2,y_3} \langle \alpha_i^{y_1} \alpha_j^{y_2} \alpha_l^{y_3} \rangle_{q(\mathrm{y}_h)} f_k(y_1,y_2,y_3,\mathrm{x},\zeta) \end{array} \right\}. \tag{4}$$

Normalization will lead to the desired probabilities μ_{il}.

Similarly, keep $q(\mathrm{s})$ fixed and we get

$$KL(q_0||p) = -\langle \log p(\mathrm{s}, \mathrm{y}_h, \mathrm{y}_e|\mathrm{x}) \rangle_{q_0} - H(q(\mathrm{y}_h)) + c',$$

where c' is another constant. Let the derivative over m_i^y equal zero, and we get

$$m_i^y \propto \exp \sum_k \lambda_k \sum_{j l y_1 y_2} \left\{ \begin{array}{l} n_{ij} \langle s_{il} s_{jl} \rangle_{q(\mathrm{s})} \langle \alpha_j^{y_1} \alpha_l^{y_2} \rangle_{q(\mathrm{y}_h)} f_k(y,y_1,y_2,\mathrm{x},\zeta) + \\ n_{ij} \langle s_{jl} s_{il} \rangle_{q(\mathrm{s})} \langle \alpha_j^{y_1} \alpha_l^{y_2} \rangle_{q(\mathrm{y}_h)} f_k(y_1,y,y_2,\mathrm{x},\zeta) + \\ n_{jl} \langle s_{ji} s_{li} \rangle_{q(\mathrm{s})} \langle \alpha_j^{y_1} \alpha_l^{y_2} \rangle_{q(\mathrm{y}_h)} f_k(y_1,y_2,y,\mathrm{x},\zeta) \end{array} \right\}. \tag{5}$$

Note that since s_{il} and α_i^y are all indicator variables, their expectations are the marginal probabilities μ_{il} and m_i^y respectively. Also, because of the naïve mean field assumption of $q(\mathrm{s})$ and $q(\mathrm{y}_h)$, the expectations of the product of the indicator variables is the product of their corresponding marginal probabilities, that is, $\langle s_{il} s_{jl} \rangle_{q(\mathrm{s})} = \mu_{il}\mu_{jl}$, $\langle s_{ji} s_{li} \rangle_{q(\mathrm{s})} = \mu_{ji}\mu_{li}$, $\langle \alpha_j^{y_1} \alpha_l^{y_2} \rangle_{q(\mathrm{y}_h)} = m_j^{y_1} m_l^{y_2}$, and $\langle \alpha_i^{y_1} \alpha_j^{y_2} \alpha_l^{y_3} \rangle_{q(\mathrm{y}_h)} = m_i^{y_1} m_j^{y_2} m_l^{y_3}$.

Equations (4) and (5) are a set of coupled equations, also known as mean field equations. These equations are iteratively solved for a fixed point solution. Intuitively, parameters μ_{il} are updated by expected contributions from possible parents and neighbors, and similar for m_i^y. In (4) and (5), structure parameters μ_{il} depend on class label assignments, and m_i^y depend on expected structure connectivity. Thus, model structure selection is integrated with label assignment during the inference.

Now, we have presented a mean field approximation of the wake phase. To finish the sleep phase, the same mean field equations are enforced by coordinate descent alternating between observable variables Y_e and hidden variables S and Y_h. When first optimizing (5) for Y_e, the initial distribution of hidden variables are set as the optimal distribution at the end of wake phase. Then, take the optimal distribution of the former step as initial distribution of Y_e and optimize (4) and (5) to get an approximate distribution of hidden variables. For wake phase, initial distributions can be random and convergence is arrived at. But for sleep phase, a few steps are required to guarantee the improvement of CF_1^{App}.

2. $q(\mathrm{s}) = q(\mathrm{s_h}|\mathrm{s_v})q(\mathrm{s_v})$. Based on the assumption that $\mathrm{s_h}$ are deterministic and independent of $\mathrm{s_v}$, $q(\mathrm{s_h}|\mathrm{s_v})$ is an indicator function and takes all the probability one if $\mathrm{s_h}$ are the allowed connections.

Thus, all the terms in (1), (2), (4), and (5) can be calculated. The whole parameter estimation algorithm is as follows. First, apply (4) and (5) to iteratively compute the marginal probabilities of both wake and sleep phases. With the marginal probabilities, CF_1^{App} and its derivatives with respect to model parameters are calculated. Then, gradient-based optimization algorithms are applied to update model parameters. Here, we use the limited memory quasi-Newton method (Liu and Nocedal, 1989). The learning procedure is iterated until the relative change of CF_1^{App} is below some threshold. Although no guarantee exists that global optimization will be achieved, empirical studies show that this algorithm performs well.

For labeling a testing example, Equations (4) and (5) are iteratively solved with all variables free for a fixed point solution. At the end of convergence, the maximum a posterior model structure (a tree) is constructed from the probabilities μ_{il} by dynamic programming, and the most likely label assignments are found from the marginal probabilities m_i^y.

5. Implementation Details and Experimental Setup

Our experiments consist of two parts. The first part is to evaluate the performance of integrated web data extraction models compared with existing decoupled methods. The second part is to evaluate Dynamic Hierarchical Markov Random Fields (DHMRFs) compared with fixed-structured hierarchical models and Dynamic Trees (Williams and Adams, 1999). All the experiments are carried out on a real-world web data extraction task—production information extraction. In this section, we present the implementation details and the setup of our experiments. Results will be reported in the next two sections.

5.1 Features

As conditional models, DHMRFs and HCRFs can incorporate any useful feature for web data extraction. In this section, we present the types of features used in our experiments. As we shall note some of the features have been used in some existing extraction methods. However, they were mainly used as heuristic rules.

5.1.1 FEATURES OF ELEMENTS

For each element, we extract both content and visual features as listed in Table 1. All the features can be obtained through rendering a page. Previous work (Zhai and Liu, 2005; Zhu et al., 2005; Zhao et al., 2005; Gatterbauer et al., 2007) has shown the effectiveness of visual features for webpage analysis and information extraction.

5.1.2 FEATURES OF BLOCKS

The features of inner blocks are aggregations of their children's features. These features can be extracted via a bottom-up procedure starting from leaf nodes (or elements), such as the number of the children having a particular feature and the presence of a feature or a simultaneous presence of several features among the children. We also compute the following distances for each block to exploit the regularity of similar data records in a page.

Tree Distance Features: if two blocks are visually similar, usually their sub-trees on a vision-tree are also similar. We define the tree distance of two blocks as a measure of their structure similarity. The tree distance of two blocks is defined as the edit distance of their corresponding sub-

Name	Description
Content	The Content of a text element
Tag	The tag name of an element
Font Size	The font size of an element
Font Weight	The font weight of an element
Position	The coordinates of an element
Height	The height of an element's rectangle
Width	The width of an element's rectangle
Area	The area of an element's rectangle
Image URL	The source URL of an image element
Link URL	The action URL of an element if it exists
Image Alt-text	The alternative text of an image element

Table 1: The content and visual features of each element.

trees. Although the time-complexity of computing this distance could be high, we can substantially reduce the computation with some heuristics. For example, if the depth difference of two sub-trees is too large, they are not likely to be similar and this computation is not necessary. Once we have computed the tree distances, we can use some thresholds to define boolean-valued feature functions. For example, if the tree distance of two adjacent blocks is not more than 0.2, they are both likely to be data records.

Shape Distance and Type Distance Features: we also compute the shape distance and type distance (Zhao et al., 2005) of two blocks to exploit their similarity. For shape distance, we use the same definition of shape codes and the same calculation method as in the work (Zhao et al., 2005). To compute the type distance of two blocks, we define the following types for each element:

IMAGE: the element is an image.

JPEG IMAGE: the image element that is also a jpeg picture.

CODED IMAGE: the image element whose source URL contains at least three succeeding numbers, such as "/products/s_thumb/eb04iu_0190893_200t1.jpg".

TEXT: the element has text content.

LINK TEXT: the text element that contains an action URL.

DOLLAR TEXT: the text element that contains at least one dollar sign.

NOTE TEXT: the text element whose tag is "input", "select" or "option".

NULL: the default type of each element.

After defining each element's type code, a block's type code is defined as a sequence of the type codes of its children. As in the work by Zhao et al. (2005), multiple consecutive occurrences of each type are compressed to one occurrence. The edit distance of type codes is the type distance of two blocks.

Similar to the use of tree distance, we can easily incorporate shape distance and type distance by defining boolean-valued feature functions with pre-determined thresholds. Note that our model will not be sensitive to these thresholds because the defined feature functions are softened by learning a weight for each of them. Each feature function contributes its weight to the probability only when it is active. If a feature function is always active, it has no effect on the probability; and if a feature

Label Name	Semantic Meaning
Con_Image	Contains product's image
Con_Name	Contains product's name
Con_Price	Contains product's price
Con_Desc	Contains product's description
Con_ImgNam	Contains product's image and name
Con_NamPrc	Contains product's name and price
Con_ImgPrc	Contains product's image and price
Page Head	The head part of a Web page
Page Tail	The tail part of a Web page
Nav Bar	The navigation bar of a Web page
Data Region	Contains only similar data records
Data Record	Contains all the target attributes if exist
Info Block	Contains one or more data records and some additional information
Note Block	Contains no target attributes and are also not meaningful parts of a webpage

Table 2: Label spaces of inner variables for product information extraction.

function appears sparsely in the training set, smoothing techniques can be used to avoid over-fitting. Here, we use the spherical Gaussian prior to penalize the log-likelihood function during learning.

5.1.3 GLOBAL FEATURES

As described in the introduction, data records in the same webpage are always related. Based on work by Zhai and Liu (2005), we try to align the elements of two adjacent blocks in the same page and extract some global features to help attribute labeling.

For two neighboring blocks, we use the partial tree-alignment algorithm (Zhai and Liu, 2005) to align their elements. An alignment is discarded if most of the elements are not aligned. For successful alignments, the following feature is extracted.

Repeated elements are less informative: this feature is based on the observation that repeated elements in different records are more likely to be less useful, while important information such as the name of a product is not likely to repeat in the same webpage. For example, the "Add to cart" button appears in both data records as in Figure 1(a), but each record has a unique name. Currently, we just denote whether an element is repeated in different records. More complex measures like information entropy can be easily adopted. An example feature function can be defined as: if the element x_i repeatedly appears in the aligned records, it will be more likely to be labeled as Note or noise.

5.2 Label Spaces

For variables at leaf nodes, we are interested in deciding whether a leaf block (an element) is an attribute value of the object we want to extract. However, for variables at inner nodes, our interest shifts to the understanding of whether an inner block is a data record. So, we have two types of label spaces—leaf label space for variables at leaf nodes and inner label space for variables at inner nodes. The leaf label space consists of all the attribute names of the object we want to extract. In

product information extraction, the leaf label space consists of *Name*, *Image*, *Price*, *Description*, and *Note*. *Note* is used to describe the data we are not interested in.

The inner label space can be partitioned into an object-type independent part and an object-type dependent part. We explain how to define these two parts in turn:

Object-type Independent Labels: Since we want to extract data from webpages, the labels *Page Head*, *Page Tail*, *Nav Bar*, and *Info Block* are naturally needed to denote different parts of a webpage. The labels *Data Record* and *Data Region* are also required for detecting data records. The label *Note Block* is also required to denote blocks that do not contain any meaningful information, such as the attributes to be extracted and the head, tail or navigation bar of a webpage. All these labels are general to any web data extraction problem, and they are independent of any specific object type.

Object-type Dependent Labels: Between data record blocks and leaf blocks, there are intermediate blocks on a vision-tree. So, we must define some intermediate labels between *Data Record* and the labels in the leaf label space. These labels are object-type dependent because intermediate blocks contain some object specific attribute values. A natural method is to use the combinations of the attributes to define intermediate labels. Of course, if we use all the possible combinations, the label space could be too large. We can discard unimportant combinations by considering the co-occurrence frequencies of their corresponding attribute values in the training data. The object-type dependent labels in product information extraction are listed in Table 2 with the format Con_*.

5.3 Data Sets

We set up two general data sets with randomly crawled product webpages. The list data set (*LDST*) contains 771 list pages and the detail data set (*DDST*) contains 450 detail pages. All the pages are parsed by VIPS and are hierarchically labeled, that is, every block in the parsed vision-trees is labeled. We use 200 list pages and 150 detail pages to learn the parameters of different models. The remaining pages (571 list pages and 300 detail pages) are used for testing. For each product item, we want to extract four attributes—*Name*, *Image*, *Price*, and *Description*.

For the training data, the detail pages are from 61 web sites and the list pages are from 81 web sites. The number of web sites that are found in both list and detail training data is 39. Thus, in total the training pages are taken from 103 different web sites. Totally, 58 unique templates are presented in the list training pages and 61 unique templates are presented in the detail training pages. For testing data, Table 3 shows the number of unique web sites where the pages come from and the number of different templates presented in these data. For example, the pages in *LDST* are from 167 web sites, of which 78 are found in list training data and 52 are found in detail training data. The number of web sites that are found in both list and detail training data is 34. Similar interpretation applies to other numbers in the table. Thus, totally 71 list page web sites and 263 detail page web sites are not seen in the training data. For templates, 83 list page templates and 208 detail page templates are not seen the training data. For different templates, the number of documents varies. In *LDST*, most of the templates have 2 to 5 documents. In *DDST*, pages from different web sites typically have different templates and thus most templates have 1 document.

5.4 Evaluation Metrics

For data record detection, we use the standard *P*recision, *R*ecall and *F1* measure to evaluate the methods. A block is considered as a correctly detected data record if it contains all the appeared

Data Sets	*LDST*	*DDST*
#Web Site	167 (78/52/34)	268 (2/3/0)
#Template	140 (57/0/0)	212 (0/4/0)

Table 3: Statistics of the data sets.

attributes of one object, and does not contain any attributes of other objects. A correct data record could tolerate (miss or contain) some non-important information like "Add to Cart" button.

For attribute labeling, the performance on each attribute is evaluated by *P*recision (the percentage of returned elements that are correct), *R*ecall (the percentage of correct elements that are returned), and their harmonic mean *F1*. We also use two comprehensive evaluation criteria:

Block Instance Accuracy (Blk_IA): the percentage of data records of which the key attributes (*Name*, *Image*, and *Price*) are all correctly labeled.

Average F1 (Avg_F1): the average of F1 values of different attributes.

6. Evaluation of Integrated Web Data Extraction Models

In this section, we report the evaluation results of integrated web data extraction models compared with decoupled models. The results demonstrate that integrated extraction models can achieve significant improvements over decoupled models in both record detection and attribute labeling. We also show the generalization ability of the integrated extraction models.

6.1 Methods

We build the baseline methods by sequentially combining the record detection algorithm DEPTA (Zhai and Liu, 2005) and 2D CRFs (Zhu et al., 2005). For detail pages, which DEPTA cannot deal with, we first detect the main data block using the method by Song et al. (2004) and then use 2D CRFs to perform attribute labeling on the detected main block. For the integrated extraction model, a webpage is first segmented by VIPS to construct a vision-tree and then HCRFs are used to detect both records and attributes on the vision-tree. Note that all the HCRFs evaluated in this section are the model in Figure 3(b). The evaluation results of another HCRFs, which are slightly better, are presented in Section 7.

To see the effect of the global features in Section 5.1.3, we also evaluate an HCRF model that does not use these global features. We denote this model by H_NG (without global features). Similarly, we evaluate two 2D CRF models in the baseline methods. As in the work of Zhu et al. (2005), a basic 2D CRF model is set up with only the basic features (see Table 1) when labeling each detected data record. Another 2D CRF model is set up with both the basic features and the global features. We denote the basic model by 2D CRF and denote the other model by 2D_G. For 2D_G, we first cache all the detected records from one webpage and then extract the global features. As there is no tree structure here, the alignments are based on the elements' relative positions in each record.

To see the separate effect of our approach on record detection and attribute labeling, we first detect data records on the parsed vision-trees using the content features, tree distance, shape distance, and type distance features. Then, we use HCRFs to label the detected records. When doing

attribute labeling, we also evaluate two HCRF models with and without the global features. These two models are denoted by H_S and H_SNG respectively.

For all the HCRF models, we use 200 list pages and 150 detail pages together to learn their parameters. We use the same 200 list pages to train a 2D CRF model for extraction on list pages, and use the same 150 detail pages to train another 2D CRF model for extraction on detail pages. The reason for training two models for list and detail pages separately is that, for a 2D CRF model, the features and parameters for list and detail pages are quite different and a uniform model cannot work well. In the training stage, all of the algorithms converge quickly, within 20 iterations.

6.2 Results and Discussions

We compare our approach with DEPTA (Zhai and Liu, 2005) on *LDST* for data record detection. The running results of DEPTA on our data set are kindly provided by its authors. DEPTA has a similarity threshold, and it is set at 60% in this experiment. Some simple heuristics are also used in DEPTA to remove some noise records. For example, a data region that is far from the center or contains neither image nor dollar sign is removed.

6.2.1 RECORD DETECTION

The results of record detection are shown in Table 4. We can see that both HCRF and H_NG significantly outperform DEPTA in recall, improved by 8.1 points, and precision, improved by 7.5 points. The improvements come from two parts:

Advanced data representation and more features: in our model, we incorporate more features such as content features and shape distance and type distance features than DEPTA. We also adopt an advanced representation of webpages—vision-trees which have been shown to outperform tag-tree representation(Cai et al., 2004). As we can see from Table 4, H_SNG and H_S outperform DEPTA, and we gain about 2 points in precision, 7.3 points in recall, and 4.6 points in F1.

Incorporation of semantics during record detection: DEPTA just detects the blocks with regular patterns (i.e., regular tree structures) and does not take semantics into account. Thus, although some heuristics are used to remove some noise blocks, the results still contain blocks that are not data records or just parts of data records. In contrast, our approach integrates attribute labeling into block detection and can consider semantics during detecting data records. So, the blocks detected are of better quality and are more likely to be data records. For instance, a block containing a product's name, image, price and some descriptions is almost certain to be a data record, but a block containing only irrelevant information is unlikely to be a data record. The lower precisions of H_SNG and H_S demonstrate this. When not considering the semantics of the elements, H_SNG and H_S extract more noise blocks compared with H_NG or HCRF, so the precisions of record detection decrease by 5.5 points and the overall F1 measures decrease by 3.2 points.

6.2.2 ATTRIBUTE LABELING

As we can see from Table 5, our HCRF model significantly outperforms the baseline approach. On list pages, H_NG gains 18.7 points over 2D CRF in block instance accuracy and the achievements of HCRF are 13.9 points higher when compared with 2D CRF_G. On detail pages, our approach gains about 58 points over 2D CRF in block instance accuracy. The reasons for the better performance are:

Models	*H_SNG*	*H_S*	*H_NG*	*HCRF*	*DEPTA*
P	0.904	0.904	**0.959**	**0.959**	0.884
R	0.921	0.921	**0.930**	**0.930**	0.849
F1	0.912	0.912	**0.944**	**0.944**	0.866

Table 4: Record detection results of different methods on *LDST*.

Data Sets		LDST						DDST	
Models		*H_SNG*	*H_S*	*H_NG*	*HCRF*	*2D CRF*	*2D_G*	*HCRF*	*2D CRF*
P	Name	0.836	0.860	0.880	**0.911**	0.763	0.851	**0.835**	0.398
	Image	0.901	0.905	0.952	**0.966**	0.842	0.838	**0.978**	0.546
	Price	0.906	0.903	0.959	**0.963**	0.913	0.915	**0.986**	0.809
	Desc	0.783	0.766	**0.792**	0.788	0.769	0.779	**0.663**	0.588
R	Name	0.851	0.875	0.854	**0.882**	0.735	0.822	**0.761**	0.398
	Image	0.917	0.921	0.924	**0.936**	0.811	0.809	**0.892**	0.546
	Price	0.922	0.919	0.930	**0.933**	0.879	0.883	**0.899**	0.809
	Desc	**0.797**	0.780	0.768	0.764	0.741	0.752	**0.604**	0.395
F1	Name	0.843	0.867	0.867	**0.896**	0.749	0.836	**0.796**	0.398
	Image	0.909	0.913	0.938	**0.951**	0.826	0.823	**0.933**	0.546
	Price	0.914	0.911	0.944	**0.948**	0.896	0.899	**0.940**	0.809
	Desc	**0.790**	0.773	0.780	0.776	0.755	0.765	**0.632**	0.473
Avg_F1		0.864	0.866	0.882	**0.893**	0.807	0.831	**0.825**	0.556
Blk_IA		0.789	0.816	0.856	**0.890**	0.669	0.751	**0.817**	0.231

Table 5: Attribute labeling results of different methods on both *LDST* and *DDST*, where *Desc* stands for *Description*.

Attribute labeling benefits from good quality records: one reason for this better performance is that attribute labeling can benefit from the good results of record detection. For example, if a detected record is not a data record or misses some important information such as *Name*, attribute labeling will fail to find the missed information or will find some incorrect information. So, H_SNG outperforms 2D CRF and H_S outperforms 2D_G. Of course the achievements of H_SNG and H_S may also come from the incorporation of long distance dependencies, which will be discussed later.

Global features help attribute labeling: another reason for the improvements in attribute labeling is the incorporation of the global features as in Section 5.1.3. From the results, we can see that when considering global features, attribute labeling is more accurate. For example, 3.4 points are gained in block instance accuracy by HCRF compared with H_NG, and H_S achieves 2.7 points in block instance accuracy compared with H_SNG. For the two baseline methods, compared with 2D CRF, which uses only the features of the elements in each detected record, more than 8 points are gained in block instance accuracy by 2D_G, which incorporates the global features.

HCRF models incorporate long distance dependencies: the third reason is the incorporation of long distance dependencies. From the results, we can see that hierarchical models could get

promising results while 2D CRFs perform poorly on detail pages. This is because, for a detected record, 2D CRFs put its elements in a two-dimensional grid and long distance interactions cannot be incorporated in the flat model, due to the first-order Markov assumption. In contrast, HCRF models can incorporate dependencies at various levels and thus incorporate long distance dependencies. For detail pages, as there is no record detection, H_SNG and H_S are not applicable here. There are no global features either, so we just list the results of HCRF and 2D CRF in Table 5.

The quite different performance of 2D CRFs on list and detail pages says the same thing about the effectiveness of long distance dependencies. For list pages, the inputs are data records, which always contain a small number of elements. In this case, 2D CRFs can effectively model the dependencies of the attributes and achieve reasonable accuracy. Note that the results on detail pages are achieved without any pre-processing to remove noise elements. Empirical studies show that some appropriate pre-processing can improve the performance significantly on detail pages.

6.3 Generalization Ability

We report some empirical results to show the generalization ability of the integrated web data extraction models. We randomly pick 37 templates from *LDST* and for each template we collect 5 webpages for training and 10 webpages for testing. We randomly select $N (N = 1,2,3,\cdots,37)$ templates together with their training pages as training data, and test the model on all the testing webpages of the 37 templates. For each N, we run the integrated HCRFs 10 times and take the average as the final results. Figure 5 shows the average F1 and block instance accuracy against different N. We can see that the integrated extraction models converge very quickly. As the number of templates increase in the training data, the extraction accuracy becomes higher and the variances become smaller. The strong generalization ability to unseen templates is mainly due to the very general and robust visual features we are using in our models. For different templates, although the low-level HTML codes or HTML tag trees are quite different, the visual layout and visual features they use are usually common. Thus, we can learn a robust model from a small set of templates and generalize well to unseen templates. Section 7.3 presents another set of results that show the generalization ability to unseen templates.

7. Evaluation of Dynamic Hierarchical Markov Random Fields

In this section, we report the evaluation results of Dynamic Hierarchical Markov Random Fields compared with fixed-structured hierarchical models and Dynamic Trees. Results show that DHMRFs can (at least partially) overcome the blocky artifact issue in diverse web data extraction. We also present some empirical studies about the learning algorithm of DHMRFs.

7.1 Models

We compare DHMRFs with HCRFs in both Figure 3(b) and 3(c), Dynamic Trees (D-Trees), and fixed-structured tree models (F-Trees). For HCRFs and F-Trees, all training pages are hierarchically labeled. The training is complete and exact message passing algorithms are used to learn their parameters and find MAP label assignments. For DHMRFs and D-Trees, labels of leaf nodes are kept the same and inner labels are hidden during learning. For the incomplete training, we apply the variational method developed in this paper for DHMRFs. Mean field approximation is also used for Dynamic Trees. For DHMRFs and HCRFs, the same set of feature functions are used for class

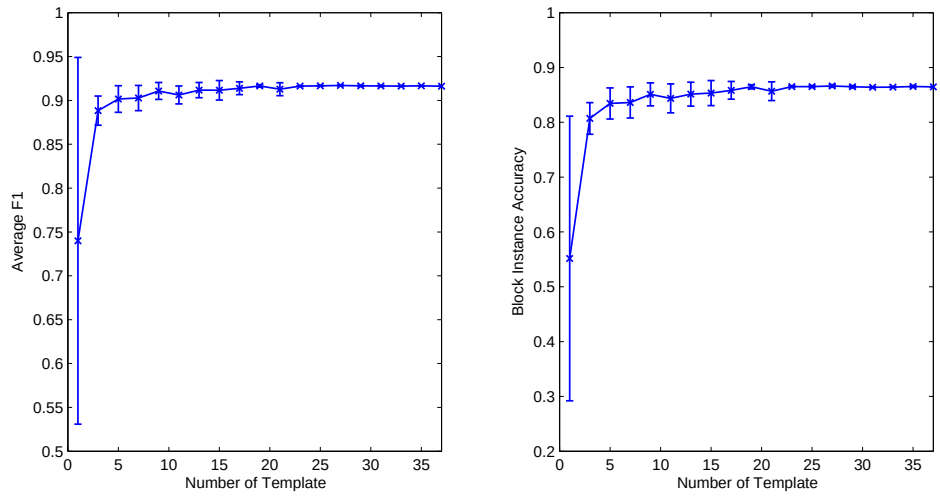


Figure 5: The left plot is the mean and variance of the Average F1 and the right plot is the mean and variance of the Block Instance Accuracy.

label assignment. We will use *HCRF* and *HCRF+* to denote the two HCRF models in Figure 3(b) and 3(c) respectively.

To apply DHMRFs and D-Trees, initial configuration of the model structure must be carried out first. Basically, we need to initially set the number of layers and the number of nodes at each layer. It may be different for different application domains to set the initial configuration. For image processing, it can be done via sub-sampling or wavelet filtering. For web data extraction, the data are represented as texts, images, buttons, and so on. These atomic information units are more expressive compared to image pixels. There is definitely no benefit to view a webpage as a collection of image pixels and then apply the methods in image processing. Here, we use the same number of layers (and the same number of nodes at each layer) in dynamic models as in the vision-trees. Note that additional nodes can be introduced. For DHMRFs feature functions can be easily defined to consider these nodes, and for D-Trees the *part-time-node-employment* prior (Adams and Williams, 1999) can be applied to get a sparse structure.

For D-Trees, two sets of parameters—conditional probability tables (CPTs) and affinities, need to be set. We keep the affinities fixed and learn the CPTs. To avoid over-parametrization, layer-wise CPT sharing is adopted in previous work. However, for heterogeneous web data, three-layer-wise sharing is better. That is, every three layers from the top down share one CPT. To incorporate evidence, we use the class-independent model (Storkey and Williams, 2003) with emission distributions set as the empirical frequencies in the training data set. CPTs are also initialized as frequencies. To avoid zero probabilities of unseen samples, Laplace's rule is used with pseudocount set at one. Our study shows that when the affinities are set as 0 for the natural parent, -1 for the nearest neighbors of the natural parent, and -3 for the null parent, better performance is achieved compared to previously used settings. The CPTs used in our experiments are achieved with 10 iterations.

7.2 Extraction Accuracy

Table 6 shows the extraction accuracy of different models. From the results, we can see that DHMRFs achieve the highest performance on both data sets. Compared to the fixed *HCRF*, on *LDST* about 3 points in Average F1 and about 5 points in Block Instance Accuracy are gained. Compared to the more complex *HCRF+*, more than 2 points in Average F1 and about 3 points in Block Instance Accuracy are achieved. More specifically, compared to *HCRF+*, more than 3 points are achieved in both precision and recall on *Name*, and more than 2 points are achieved on *Desc*. For *Image* and *Price* the improvements are smaller. This is because *Image* and *Price* are usually more distinctive than the other attributes. So both models perform quite well. On *DDST*, the improvements in *Name* are about 4 points in both precision and recall, and for *Description* the improvements are about 7 points in both precision and recall. Small improvements are achieved in *Image* and *Price* due to the same reason as in list pages.

The improvements demonstrate the merits of DHMRFs. First, DHMRFs can incorporate the two-dimensional neighborhood dependencies among the nodes at the same level, which have been shown to be useful (Zhu et al., 2005). The better performance of *HCRF+* compared to *HCRF* also shows the usefulness of two-dimensional neighborhood dependencies. By dynamically selecting connections between different nodes, DHMRFs can bring together the attributes of the same object (here, an object is a product item), and thus the correlation between these attributes can be strengthened. Second, DHMRFs can deal with webpages with intertwined attributes (Zhai and Liu, 2005). For these webpages, the attributes of different objects are intertwined in HTML tag trees. Unaware of semantic labels, the constructed vision-trees also have intertwined attributes. In these cases, fixed-structured HCRFs (both *HCRF* and *HCRF+*) cannot correctly detect data records by simply assigning labels to the nodes of a vision-tree. Instead, as structure selection is integrated with labeling in DHMRFs, the dynamic model can properly group the attributes of the same object, and at the same time separate the attributes of different objects with the help of semantic labels. The semantic labels have been shown to be helpful in detecting data records (i.e., groups of attributes) in previous experiments. Note that although intertwined cases are usually fewer than non-intertwined cases, they are not sparse samples in our model. This is because although their edge connections in HTML tag trees are somewhat different from non-intertwined ones, the visual features they share are almost the same. Thus, training samples with or without intertwined cases can teach a good model. In fact, to keep it fair for both dynamic models and fix-structured models, we only provide non-intertwined samples during training.

Compared to the fixed F-Trees, the worse performance of D-Trees is quite counter-intuitive. However, a close examination of the results reveals that the reason for the worse performance is due to the less discriminative power of D-Trees. As we have stated, for diverse web data CPT sharing can be difficult. Although empirical studies can find a good sharing method, we couldn't learn an optimal model with a limited set of training samples. Furthermore, its generative characteristic causes difficulty in encoding useful features. In this way, more uncertainty in structure selection couldn't be resolved than that in DHMRFs. This is evident if we look at the average log-likelihood of the MAP connections over all samples and all nodes. For D-Trees the average value is -0.4080, and for DHMRFs it is -0.3170. In terms of probability, they are equivalent to 0.6650 and 0.7283 respectively. The less discriminative power of D-Trees causes additional errors in constructing model structures even for the non-intertwined cases, and thus hurts the accuracy of record detection and attribute labeling. So, D-Trees perform worse than F-Trees, which can deal with the non-

Data Sets		LDST					DDST				
Models		*F-Tree*	*D-Tree*	*HCRF*	*HCRF+*	*DHMRF*	*F-Tree*	*D-Tree*	*HCRF*	*HCRF+*	*DHMRF*
P	Name	0.890	0.879	0.911	0.920	**0.952**	0.829	0.785	0.835	0.835	**0.874**
	Image	0.959	0.951	0.966	0.968	**0.988**	0.972	0.928	**0.978**	**0.978**	**0.978**
	Price	0.960	0.937	0.963	0.972	**0.978**	0.976	0.947	0.986	**0.990**	0.989
	Desc	0.804	0.800	0.788	0.805	**0.828**	0.722	0.698	0.663	0.656	**0.730**
R	Name	0.842	0.744	0.882	0.897	**0.928**	0.779	0.684	0.761	0.753	**0.799**
	Image	0.908	0.805	0.936	0.944	**0.958**	0.868	0.809	0.892	0.883	**0.898**
	Price	0.910	0.794	0.936	**0.951**	0.949	0.888	0.826	0.899	0.893	**0.905**
	Desc	0.762	0.678	0.764	0.786	**0.811**	0.641	0.609	0.604	0.603	**0.668**
F1	Name	0.865	0.806	0.896	0.908	**0.940**	0.803	0.731	0.796	0.792	**0.835**
	Image	0.933	0.872	0.951	0.956	**0.973**	0.917	0.864	0.933	0.928	**0.936**
	Price	0.934	0.860	0.948	0.961	**0.963**	0.930	0.882	0.940	0.939	**0.945**
	Desc	0.782	0.734	0.776	0.795	**0.819**	0.679	0.650	0.632	0.628	**0.698**
Avg_F1		0.879	0.818	0.893	0.902	**0.924**	0.832	0.782	0.825	0.822	**0.854**
Blk_IA		0.869	0.837	0.890	0.912	**0.940**	0.809	0.762	0.817	0.819	**0.853**

Table 6: Extraction accuracy on *LDST* and *DDST*, where *Desc* stands for *Description*.

intertwined cases well. The results also show that the directed tree models can perform well on our data sets, but are inferior to HCRFs.

7.3 Extraction Accuracy on Unseen Templates

For detail pages, since only a small number (i.e., 4) of templates in the testing data are seen in the training data, the results on webpages generated from unseen templates do not change much. Here, we only report the results on list pages. In total, *LDST* has 83 templates that are not seen in the training data. We select out all the pages with unseen templates, the total number being 190. Figure 6 shows the results of our models on these webpages. The overall performance is still very promising although it is lower than that on the whole set of webpages. Generally, the Dynamic Hierarchical Markov Random Fields always outperform all the other models. The integrated HCRFs outperform the sequential HCRFs, which take record detection and attribute labeling as two separate steps as described in Section 6.1. Dynamic Trees achieved the worst results due to the same reason of a less discriminative power in structure selection.

7.4 Fitness of Model Structure

Figure 7(a) compares the posterior probabilities of the MAP structures constructed by DHMRFs with those of the fixed structures. In terms of the number of nodes, the sizes of webpages change from 39 to 576 (average 166) in *LDST*, and the log posteriors change from -503.80 to -4.49 (average -50.7). In *DDST*, sizes range from 14 to 705 (average 131), and log posteriors range from -184.40 to -1.72 (average -42.47). Here, we only present the samples whose log posteriors are between -200 and 0 because most of the samples ($>$ 97%) fall into this interval. We can see that the MAP structures by DHMRFs always appear above the equal probability line. Thus, the structures found by the dynamic

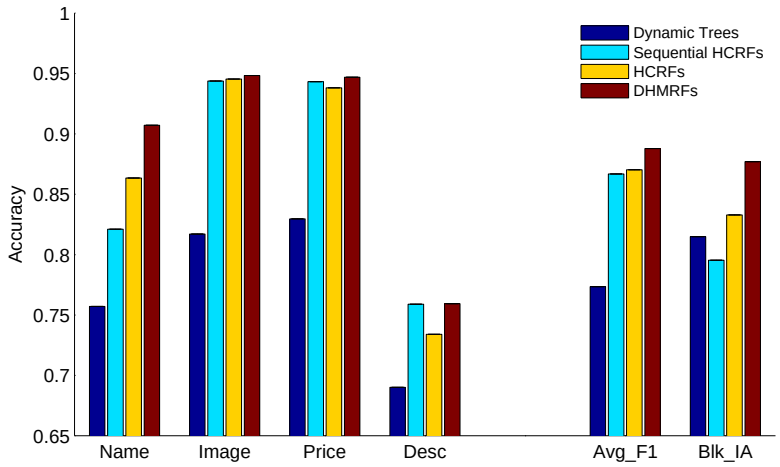


Figure 6: The performance of Dynamic Trees, Sequential HCRFs, HCRFs, and DHMRFs on the webpages whose templates are not presented in the training data. From left to right, the first four groups of the columns are the F1 of different attributes.

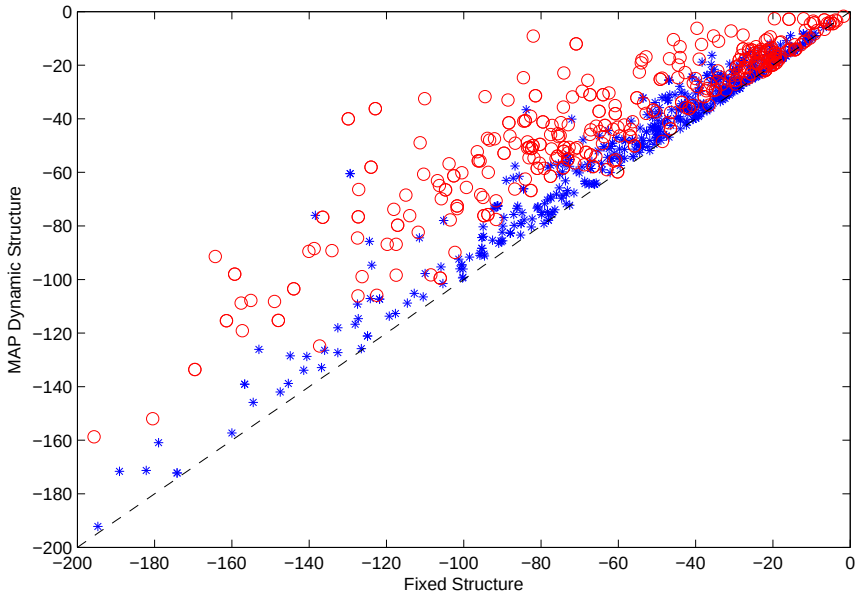
model have higher posterior probabilities. Another observation is that the distribution of samples from *DDST* is more disperse than that of the samples from *LDST*. The reason is that in list pages the attributes of an object always concentrate into small clusters, while they can scatter anywhere in detail pages.

7.5 Study about the Inference Algorithm

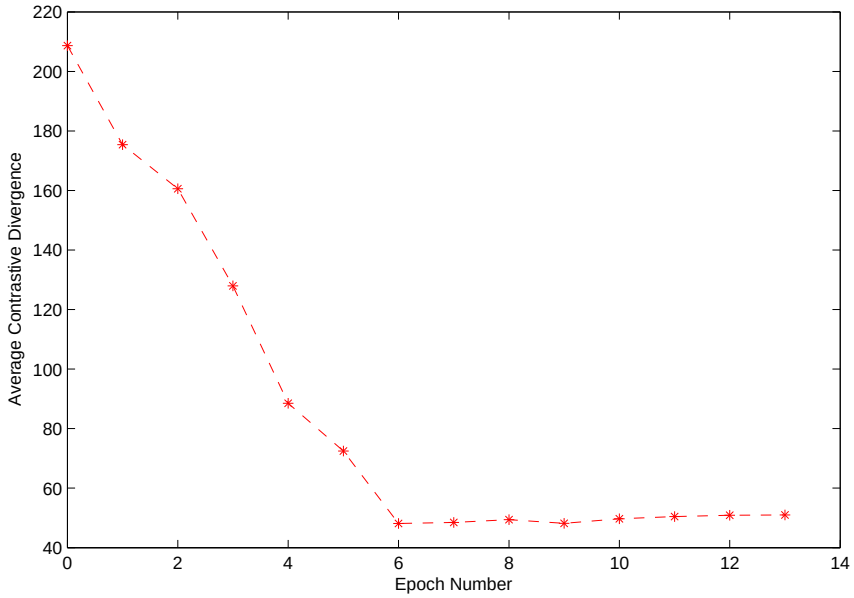
Figure 7(b) shows the change of average contrastive divergence with respect to iteration numbers in the learning of DHMRFs. To initialize the algorithm, at the wake phrase m_i^y are set to a uniform distribution plus a Gaussian noise with zero mean and variance 0.01, and μ_{il} are set to a random distribution. The model weights are initialized to zero. We can see that before 7 iterations average contrastive divergence decreases stably. And after 7, slight disturbances appear. But as for extraction accuracy, marginal changes occur (no more than 0.5 point in Block Instance Accuracy). Thus, the learning algorithm is quite stable. All the above results are achieved at iteration 7. The same initialization is used in labeling, and by running both learning and labeling many times, we observe that the algorithm is insensitive to the random initialization. Since the mean field equations are locally calculated and their update can typically converge within 5 iterations, both the learning and labeling are efficient.

8. Conclusions and Future Work

In this paper, we proposed an integrated web data extraction paradigm with hierarchical models. The proposed model is called Dynamic Hierarchical Markov Random Fields (DHMRFs), which take fixed-structured Hierarchical Conditional Random Fields (HCRFs) as a special case. DHMRFs incorporate structural uncertainty in a discriminative manner. By dynamically selecting connections



(a)



(b)

Figure 7: (a) The log posteriors of MAP dynamic structures against those of fixed structures. Samples in asterisks are from *LDST* and those in circles are from *DDST*; (b) The change of average contrastive divergence with respect to iteration numbers.

between variables, DHMRFs can potentially address the blocky artifact issue in diverse web data extraction. Compared to directed models, DHMRFs are compact in representation and powerful in encoding useful features. We develop a contrastive divergence learning algorithm to learn the parameters for DHMRFs. For the special case—HCRFs, parameter learning can be exactly performed with some assumption about the linearity of the neighborhood dependencies among sibling nodes, and without such an assumption piecewise learning can be applied to achieve a good approximation. We apply the models to a real-world web data extraction task. Experimental results show that: (1) integrated extraction models perform significantly better than decoupled methods on both record detection and attribute labeling; (2) DHMRFs can potentially address the blocky artifact issue in diverse web data extraction; (3) integrated extraction models can generalize well to unseen templates.

In our experiments, we apply a simple method to select labels for inner variables according to the co-occurrence frequency. Apparently, labels should not be selected independently and methods considering the correlations between different labels could be more desirable. We plan to try advanced methods in the future. It is also interesting to develop models that can automatically discover the number of layers and the number of nodes at each layer. Finally, extensive studies of the integrated extraction models in other complicated domains, like extracting researchers' information (Zhu et al., 2007a), is also to comprise our future work.

Acknowledgments

We thank the anonymous reviewers for helpful comments in improving the earlier version of the paper. The authors Jun Zhu and Bo Zhang are supported by National Natural Science Foundation of China under the Grant No. 60621062, and National Key Foundation R&D Project under the Grant No. 2003CB317007 and 2004CB318108.

References

Nicholas J. Adams and Christopher K. I. Williams. SDTs: Sparse dynamic trees. In *Artificial Neural Networks*, 1999.

Arwind Arasu and Hector Garcia-Molina. Extracting structured data from webpages. In *Proc. of the International Conference on Management of Data*, San Diego, CA, 2003.

David Buttler, Ling Liu, and Calton Pu. A fully automated object extraction system for the world wide web. In *Proc. of International Conference on Distributed Computing Systems*, Arizona, USA, 2001.

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Block-based web search. In *Proc. of the Internaltinoal Conference on Information Retrieval*, Sheffield, UK, 2004.

Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On contrastive divergence learning. In *Proc. of Artificial Intelligence and Statistics*, Barbados, 2005.

Chia-Hui Chang and Shao-Chen Lui. IEPAD: Information extraction based on pattern discovery. In *Proc. of the International World Wide Web Conference*, Hong Kong, China, 2001.

Robert G. Cowell, A.Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, NY, 1999.

Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. ROADRUNNER: Towards automatic data extraction from large web sites. In *Proc. of the Conference on Very Large Data Bases*, Rome, Italy, 2001.

Aron Culotta, Trausti Kristjansson, Andrew McCallum, and Paul Viola. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence Journal*, 170(14):1101–1122, 2006.

David W. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents. In *Proc. of the International Conference on Management of Data*, Philadephia, PA, 1999.

Wolfgang Gatterbauer, Paul Bohunsky, Marcus Herzog, Bernhard Krupl, and Bernhard Pollak. Towards domain-independent information extraction from web tables. In *Proc. of the International World Wide Web Conference*, Banff, Canada, 2007.

Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of relational structure. In *Proc. of the International Conference on Machine Learning*, Williams College, Williamstown, MA, 2001.

Xuming He, Richard S. Zemel, and Miguel A. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, 2004.

Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

William W. Irving, Paul W. Fieguth, and Alan S. Willsky. An overlapping tree approach to multiscale stochastic modeling and estimation. *IEEE Trans. on Image Processing*, 6(11):1517–1529, 1997.

Michael I. Jordan, Zoubin Ghahramani, Tommis Jaakkola, and Lawrence K. Saul. *An Introduction to Variational Methods for Graphical Models*. M. I. Jordan (Ed.), Learning in Graphical Models, Cambridge: MIT Press, Cambridge, MA, 1999.

Zoltan Kato, Marc Berthod, and Josiane Zerubia. Multiscale Markov random field models for parallel image classification. In *IEEE International Conference on Computer Vision*, Berlin, Germany, 1993.

Sanjiv Kumar and Martial Hebert. A hierarchical field framework for unified context-based classification. In *IEEE International Conference on Computer Vision*, Beijing, China, 2005.

Nicholas Kushmerick. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning*, Williams College, Williamstown, MA, 2001.

Kristina Lerman, Lise Getoor, Steven Minton, and Craig Knoblock. Using the structure of web sites for automatic segmentation of tables. In *Proc. of the International Conference on Management of Data*, Paris, France, 2004.

Jia Li, Robert M. Gray, and Richard A. Olshen. Multiresolution image classification by hierarchical modeling with two-dimensional hidden Markov models. *IEEE Trans. on Information Theory*, 46 (5):1826–1841, 2000.

Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition. In *Advances in Neural Information Processing Systems*, Whistler, Canada, 2005.

Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semi-structured information sources. *Journal of Autonomous Agents and Multi-Agent*, 4(1-2):93–114, 2001.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2004.

Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph ($p^\star$) model for social networks. *Social Networks*, 2006.

Ruihua Song, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In *Proc. of the International World Wide Web Conference*, Budapest, Hungary, 2004.

Amos J. Storkey and Christopher K. I. Williams. Image modeling with position-encoding dynamic trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(7):859–871, 2003.

Charles Sutton and Andrew McCallum. Piecewise training for undirected models. In *Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, 2005.

Charles Sutton, Khashayar Rohanimanesh, and Andrew McCallum. Dynamic conditional random fields: factorized probabilistic models for labeling and segmenting sequence data. In *Proc. of the International Conference on Machine Learning*, Banff, Canada, 2004.

Sinisa Todorovic and Michael C. Nechyba. Dynamic trees for unsupervised segmentation and matching of image regions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(11): 1762–1777, 2005.

Martin Wainwright, Tommi Jaakkola, and Alan Willsky. A new class of upper bounds on the log partition function. In *Uncertainty in Artificial Intelligence*, Alberta, Canada, 2002.

Max Welling and Geoffrey E. Hinton. A new learning algorithm for mean field boltzmann machines. In *International Conference on Artificial Neural Networks*, Vienna, Austria, 2001.

Max Welling and Charles Sutton. Learning in Markov random fields with contrastive free energies. In *Artificial Intelligence and Statistics*, Barbados, 2005.

Christopher K. I. Williams and Nicholas J. Adams. DTs: dynamic trees. In *Advances in Neural Information Processing Systems*, Denver, Colorado, USA, 1999.

Alan S. Willsky. Multiresolution Markov models for signal and image processing. In *Proc. of the IEEE*, 2002.

Alan L. Yuille. The convergence of contrastive divergence. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2004.

Yanhong Zhai and Bing Liu. Web data extraction based on partial tree alignment. In *Proc. of the International World Wide Web Conference*, Chiba, Japan, 2005.

Hongkun Zhao, Weiyi Meng, Zonghuan Wu, Vijay Raghavan, and Clement Yu. Fully automatic wrapper generation for search engines. In *Proc. of the International World Wide Web Conference*, Chiba, Japan, 2005.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. 2D conditional random fields for web information extraction. In *Proc. of the International Conference on Machine Learning*, Bonn, Germany, 2005.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Wei-Ying Ma. Simultaneous record detection and attribute labeling in web data extraction. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, 2006.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Hsiao-Wuen Hon. Webpage understanding: an integrated approach. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, San Jose, CA, 2007a.

Jun Zhu, Zaiqing Nie, Bo Zhang, and Ji-Rong Wen. Dynamic hierarchical Markov random fields and their application to web data extraction. In *Proc. of the International Conference on Machine Learning*, Corvallis, OR, 2007b.

Journal of Machine Learning Research 9 (2008) 1615-1646 Submitted 7/07; Revised 1/08; Published 7/08

Universal Multi-Task Kernels

Andrea Caponnetto CAPONNET@CITYU.EDU.HK
Department of Mathematics
City University of Hong Kong
83 Tat Chee Avenue, Kowloon Tong, Hong Kong

Charles A. Micchelli CAM@MATH.ALBANY.EDU
Department of Mathematics and Statistics
State University of New York
The University at Albany
Albany, New York 12222, USA

Massimiliano Pontil M.PONTIL@CS.UCL.AC.UK
Department of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK

Yiming Ying ENXYY@BRIS.AC.UK
Department of Engineering Mathematics
University of Bristol
Queen's Building, Bristol, BS8 1TR, UK

Editor: Bernhard Schoelkopf

Abstract

In this paper we are concerned with reproducing kernel Hilbert spaces $\mathcal{H}_K$ of functions from an input space into a Hilbert space $\mathcal{Y}$, an environment appropriate for multi-task learning. The reproducing kernel K associated to $\mathcal{H}_K$ has its values as operators on $\mathcal{Y}$. Our primary goal here is to derive conditions which ensure that the kernel K is universal. This means that on every compact subset of the input space, every continuous function with values in $\mathcal{Y}$ can be uniformly approximated by sections of the kernel. We provide various characterizations of universal kernels and highlight them with several concrete examples of some practical importance. Our analysis uses basic principles of functional analysis and especially the useful notion of vector measures which we describe in sufficient detail to clarify our results.

Keywords: multi-task learning, multi-task kernels, universal approximation, vector-valued reproducing kernel Hilbert spaces

1. Introduction

The problem of studying representations and methods for learning vector-valued functions has received increasing attention in Machine Learning in the recent years. This problem is motivated by several applications in which it is required to estimate a vector-valued function from a set of input/output data. For example, one is frequently confronted with situations in which multiple supervised learning tasks must be learned simultaneously. This problem can be framed as that of learning a vector-valued function $f = (f_1, f_2, \ldots, f_n)$, where each of its components is a real-valued function and corresponds to a particular task. Often, these tasks are dependent on each other in

that they share some common underlying structure. By making use of this structure, each task is easier to learn. Empirical studies indicate that one can benefit significantly by learning the tasks simultaneously as opposed to learning them one by one in isolation (see, e.g., Evgeniou et al., 2005, and references therein).

In this paper, we build upon the recent work of Micchelli et al. (2006) by addressing the issue of universality of multi-task kernels. Multi-task kernels were recently discussed in Machine Learning context by Micchelli and Pontil (2005), however there is an extensive literature on multi-task kernels as there are important both in theory and practice (see Amodei, 1997; Burbea and Masani, 1984; Caponnetto and De Vito, 2006; Carmeli et al., 2006; Devinatz, 1960; Lowitzsh, 2005; Reisert and Burkhardt, 2007; Vazquez and Walter, 2003, and references therein for more information)

A multi-task kernel K is the reproducing kernel of a Hilbert space of functions from an input space $\mathcal{X}$ which takes values in a Hilbert space $\mathcal{Y}$. For example, in the discussion above, $\mathcal{Y} = \mathbb{R}^n$. Generally, the kernel K is defined on $\mathcal{X} \times \mathcal{X}$ and takes values as an *operator* from $\mathcal{Y}$ to itself.[1] When $\mathcal{Y}$ is n-dimensional, the kernel K takes values in the set of $n \times n$ matrix. The theory of reproducing kernel Hilbert spaces (RKHS) as described in Aronszajn (1950) for scalar-valued functions has extensions to any vector-valued $\mathcal{Y}$. Specifically, the RKHS is formed by taking the closure of the linear span of *kernel sections* $\{K(\cdot,x)y,\ x \in \mathcal{X},\ y \in \mathcal{Y}\}$, relative to the RKHS norm. We emphasize here that this fact is fundamentally tied to a norm induced by K and is generally non-constructive. Here, we are concerned with conditions on the kernel K which ensure that all continuous functions from $\mathcal{X}$ to $\mathcal{Y}$ can be uniformly approximated on any compact subset of $\mathcal{X}$ by the linear span of kernel sections.

As far as we are aware, the first paper which addresses this question in Machine Learning literature is Steinwart (2001). Steinwart uses the expression *universal kernel* and we follow that terminology here. The problem of identifying universal kernels was also discussed by Poggio et al. (2002). One of us was introduced to this problem in a lecture given at City University of Hong Kong by Zhou (2003). Subsequently, some aspects of this problem were treated in Micchelli et al. (2003) and Micchelli and Pontil (2004) and then in detail in Micchelli et al. (2006).

The question of identifying universal kernels has a practical basis. We wish to learn a *continuous* target function $f : \mathcal{X} \to \mathcal{Y}$ from a finite number of samples. The learning algorithm used for this purpose should be consistent. That is, as the samples size increases, the discrepancy between the target function and the function learned from the data should tend to zero. Kernel-based algorithms (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004) generally use the representer theorem and learn a function in the linear span of kernel sections. Therefore, here we interpret consistency to mean that, for *any compact* subset $\mathcal{Z}$ of the input space $\mathcal{X}$ and every continuous target function $f : \mathcal{X} \to \mathcal{Y}$, the discrepancy between the target function and the learned function goes to zero uniformly on $\mathcal{Z}$ as the sample size goes to infinity. It is important to keep in mind that our input space is *not* assumed to be compact. However, we do assume that it is a Hausdorff topological space so that there is an abundance of compact subsets, for example any finite subset of the input space is compact.

Consistency in the sense we described above is important in order to study the statistical performance of learning algorithms based on RKHS. For example, Chen et al. (2004) and Steinwart et al. (2006) studied statistical analysis of soft margin SVM algorithms, Caponnetto and De Vito (2006) gave a detailed analysis of the regularized least-squares algorithm over vector-valued RKHS and

1. Sometimes, such a kernel is called operator-valued or matrix-valued kernel if $\mathcal{Y}$ is infinite of finite dimensional, respectively. However, for simplicity sake we adopt the terminology multi-task kernel throughout the paper.

proved universal consistency of this algorithm assuming that the kernel is universal and fulfills the additional condition that the operators $K(x,x)$ have finite trace. The results in these papers imply universal consistency of kernel-based learning algorithms when the considered kernel is universal. One more interesting application of universal kernels is described in Gretton et al. (2006).

This paper is organized as follows. In Section 2, we review the basic definition and properties of multi-task kernels, define the notion of universal kernel and describe some examples. In Section 3, we introduce the notion of feature map associated to a multi-task kernel and show its relevance to the question of universality. The main result in this section is Theorem 4, which establishes that the closure of the RKHS in the space of continuous functions is the same as the closure of the space generated by the feature map. The importance of this result is that universality of a kernel can be established directly by considering its features. In Section 4 we provide an alternate proof of Theorem 4 which uses the notion of vector measures and discuss ancillary results useful for several concrete examples of some practical importance highlighted in Section 5.

name	notation	information
input space	$\mathcal{X}$	a Hausdorff topological space
	$\mathcal{Z}$	compact subset of $\mathcal{X}$
	x,t,z	elements of $\mathcal{Z}$
	$\mathscr{B}(\mathcal{Z})$	Borel σ-algebra of $\mathcal{Z}$
	ν	signed scalar measure
	μ	vector measure, see Def. 8
	p,q	indices running from 1 to n
	i,j	indices running from 1 to m
output space	$\mathcal{Y}$	Hilbert space, with inner product $(\cdot,\cdot)_{\mathcal{Y}}$
	$\mathcal{B}_1$	unit ball centered at the origin, in $\mathcal{Y}$
feature space	$\mathcal{W}$	Hilbert space with inner product $\langle\cdot,\cdot\rangle_{\mathcal{W}}$
	$\mathcal{L}(\mathcal{Y},\mathcal{W})$	all bounded linear operators from $\mathcal{Y}$ into $\mathcal{W}$
	$\mathcal{L}(\mathcal{Y})$	all bounded linear operators from $\mathcal{Y}$ into itself
	A,B	elements of $\mathcal{L}(\mathcal{Y})$
	$\mathcal{L}_+(\mathcal{Y})\subseteq\mathcal{L}(\mathcal{Y})$	subset of positive linear operators
multi-task kernel	K	a function from $\mathcal{X}\times\mathcal{X}$ to $\mathcal{L}(\mathcal{Y})$, see Def. 1
	$\mathcal{H}_K$	reproducing kernel Hilbert space of K
feature representation	Φ	mapping from $\mathcal{X}$ to $\mathcal{L}(\mathcal{Y},\mathcal{W})$
	$\mathcal{C}(\mathcal{Z},\mathcal{Y})$	space of continuous $\mathcal{Y}$-valued functions on $\mathcal{Z}$
	ι	isometric mapping from $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ to $\mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$
	$\mathcal{C}_K(\mathcal{Z},\mathcal{Y})$	subset of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ generated by K, see Eq. (2)
	$\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$	subset of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ generated by Φ, see Eq. (9)

Table 1: Notation.

2. RKHS of Vector-Valued Functions

In this section, we review the theory of reproducing kernels for Hilbert spaces of vector-valued functions as in Micchelli and Pontil (2005) and introduce the notion of universal kernels.

We begin by introducing some notation. We let $\mathcal{Y}$ be a Hilbert space with inner product $(\cdot,\cdot)_{\mathcal{Y}}$ (we drop the subscript $\mathcal{Y}$ when confusion does not arise). The vector-valued functions will take values on $\mathcal{Y}$. We denote by $\mathcal{L}(\mathcal{Y})$ the space of all bounded linear operators from $\mathcal{Y}$ into itself, with the operator norm $\|A\| := \sup_{\|y\|=1} \|Ay\|$, $A \in \mathcal{L}(\mathcal{Y})$ and by $\mathcal{L}_+(\mathcal{Y})$ the set of all bounded, positive semi-definite linear operators, that is, $A \in \mathcal{L}_+(\mathcal{Y})$ provided that, for any $y \in \mathcal{Y}$, $(y,Ay) \geq 0$. We also denote, for any $A \in \mathcal{L}(\mathcal{Y})$, by A^* its adjoint. Finally, for every $m \in \mathbb{N}$, we define $\mathbb{N}_m = \{1,\ldots,m\}$. Table 1 summarizes the notation used in paper.

Definition 1 *We say that a function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ is a multi-task kernel on $\mathcal{X}$ if $K(x,t)^* = K(t,x)$ for any $x,t \in \mathcal{X}$, and it is* positive semi-definite, *that is, for any $m \in \mathbb{N}$, $\{x_j : j \in \mathbb{N}_m\} \subseteq \mathcal{X}$ and $\{y_j : j \in \mathbb{N}_m\} \subseteq \mathcal{Y}$ there holds*

$$\sum_{i,j \in \mathbb{N}_m} (y_i, K(x_i,x_j)y_j) \geq 0. \tag{1}$$

For any $t \in \mathcal{X}$ and $y \in \mathcal{Y}$, we introduce the mapping $K_t y : \mathcal{X} \to \mathcal{Y}$ defined, for every $x \in \mathcal{X}$ by $(K_t y)(x) := K(x,t)y$. In the spirit of Moore-Aronszjain's theorem, there is a one-to-one correspondence between the kernel K with property (1) and an RKHS of functions $f : \mathcal{X} \to \mathcal{Y}$ (Aronszajn, 1950), see also Micchelli and Pontil (2005) and Carmeli et al. (2006).

Throughout this paper, we assume that the kernel K is *continuous* relative to the operator norm on $\mathcal{L}(\mathcal{Y})$. We now return to the formulation of the definition of universal kernel. For this purpose, we recall that $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ is the Banach space of continuous $\mathcal{Y}$-valued continuous function on a compact subset $\mathcal{Z}$ of $\mathcal{X}$ with the *maximum norm*, defined by $\|f\|_{\infty,\mathcal{Z}} := \sup_{x \in \mathcal{Z}} \|f(x)\|_{\mathcal{Y}}$. We also define, for every multi-task kernel K, the subspace of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$

$$\mathcal{C}_K(\mathcal{Z},\mathcal{Y}) := \overline{\text{span}}\{K_x y : x \in \mathcal{Z}, y \in \mathcal{Y}\}, \tag{2}$$

where the closure is relative to the norm in the space $\mathcal{C}(\mathcal{Z},\mathcal{Y})$.

Definition 2 *We say that a multi-task kernel K is a* universal kernel *if, for any compact subset $\mathcal{Z}$ of $\mathcal{X}$, $\mathcal{C}_K(\mathcal{Z},\mathcal{Y}) = \mathcal{C}(\mathcal{Z},\mathcal{Y})$.*

In the special case that $\mathcal{Y} = \mathbb{R}^n$, the kernel function K takes values as $n \times n$ matrices. The corresponding matrix elements can be identified by the formula

$$(K(x,t))_{pq} = \langle K_x e_p, K_t e_q \rangle_K, \ \ \forall x,t \in \mathcal{X},$$

where e_p, e_q are the standard coordinate basis in $\mathbb{R}^n$, for $p,q \in \mathbb{N}_n$.

In order to describe some of the examples of multi-task kernels below, it is useful to first present the following generalization of Schur product of scalar kernels to matrix-valued kernels. For this purpose, for any $i \in \mathbb{N}_m$ we let $y_i = (y_{1i}, y_{2i}, \ldots, y_{ni}) \in \mathbb{R}^n$, so that Equation (1) is equivalent to

$$\sum_{i,j \in \mathbb{N}_m} \sum_{p,q \in \mathbb{N}_n} y_{pi} (K(x_i,x_j))_{pq} y_{qj} \geq 0. \tag{3}$$

From the above observation, we conclude that K is a kernel if and only if $\big((K(x_i,x_j)_{p,q}\big)$ as the matrix with row index $(p,i) \in \mathbb{N}_n \times \mathbb{N}_m$ and column index $(q,j) \in \mathbb{N}_n \times \mathbb{N}_m$ is positive semi-definite. This fact makes possible, as long as the dimension of $\mathcal{Y}$ is finite, reducing the proof of some properties of operator-valued kernels to the proof of analogous properties of scalar-valued kernels; this process is illustrated by the following Proposition.

Proposition 3 *Let G and K be $n \times n$ multi-task kernels. Then, the element-wise product kernel $K \circ G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^n \times \mathbb{R}^n$ defined, for any $x,t \in \mathcal{X}$ and $p,q \in \mathbb{N}_n$, by $\big(K \circ G(x,t)\big)_{pq} := \big(K(x,t)\big)_{pq}\big(G(x,t)\big)_{pq}$ is an $n \times n$ multi-task kernel.*

Proof We have to check the positive semi-definiteness of $K \circ G$. To see this, for any $m \in \mathbb{N}$, $\{y_i \in \mathbb{R}^n : i \in \mathbb{N}_m\}$ and $\{x_i \in \mathcal{X} : i \in \mathbb{N}_m\}$ we observe that

$$\sum_{i,j \in \mathbb{N}_m} (y_i, K \circ G(x_i,x_j) y_j) = \sum_{p,i} \sum_{q,j} y_{pi} y_{qj} \big(K(x_i,x_j)\big)_{pq} \big(G(x_i,x_j)\big)_{pq}. \tag{4}$$

By inequality (3), it follows that the matrix $\big(\big(K(x_i,x_j)\big)_{pq}\big)$ is positive semi-definite as the matrix with (p,i) and (q,j) as row and column indices respectively, and so is $\big(\big(G(x_i,x_j)\big)_{pq}\big)$. Applying the Schur Lemma (Aronszajn, 1950) to these matrices implies that Equation (4) is nonnegative, and hence proves the assertion. ■

We now present some examples of multi-task kernels. They will be used in Section 5 to illustrate the general results in Sections 3 and 4.

The first example is adapted from Micchelli and Pontil (2005).

Example 1 *If, for every $j \in \mathbb{N}_m$ the function $G_j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a scalar kernel and $B_j \in \mathcal{L}_+(\mathcal{Y})$, then the function*

$$K(x,t) = \sum_{j \in \mathbb{N}_m} G_j(x,t) B_j, \quad \forall x,t \in \mathcal{X} \tag{5}$$

is a multi-task kernel.

The operators B_j model smoothness across the components of the vector-valued function. For example, in the context of multi-task learning (see, e.g., Evgeniou et al., 2005, and references therein), we set $\mathcal{Y} = \mathbb{R}^n$, hence B_j are $n \times n$ matrices. These matrices model the relationships across the tasks. Evgeniou et al. (2005) considered kernels of the form (5) with $m = 2$, B_1 a multiple of the identity matrix and B_2 a low rank matrix. A specific case for $\mathcal{X} = \mathbb{R}^d$ is

$$(K(x,t))_{p,q} = \lambda x \cdot t + (1-\lambda)\delta_{pq}(x \cdot t)^2, \quad p,q \in \mathbb{N}_n,$$

where $x \cdot t$ is the standard inner product in $\mathbb{R}^d$ and $\lambda \in [0,1]$. This kernel has an interesting interpretation. Using only the first term on the right hand side of the above equation ($\lambda = 1$) corresponds to learning all tasks as the same task, that is, all components of the vector-valued function $f = (f_1, \ldots, f_n)$ are the same function, which will be a linear function since the kernel G_1 is linear. Whereas, using only the second term ($\lambda = 0$) corresponds to learning independent tasks, that is, the components of the function f will be generally different functions. These functions will be quadratic since G_2 is a quadratic polynomial kernel. Thus, the above kernel combines two heterogeneous kernels to form a more flexible one. By choosing the parameter λ appropriately, the learning model can be tailored to the data at hand.

We note that if K is a diagonal matrix-valued kernel, then each component of a vector-valued function in the associated RKHS of K can be represented, independently of the other components, as a function in the RKHS of a scalar kernel. However, in general, a multi-task kernel will not be diagonal and, more importantly, *will not* be reduced to a diagonal one by linearly transforming the

output space. For example, the kernel in Equation (5) cannot be reduced to a diagonal kernel, unless all the matrices B_j, $j \in \mathbb{N}_m$ can all be simultaneously transformed into a diagonal matrix. Therefore, in general, the component functions share some underlying structure which is reflected by the choice of the kernel and cannot be treated as independent objects. This fact is further illustrated by the next example.

Example 2 *If $\mathcal{X}_0$ is a compact Hausdorff space, for $p \in \mathbb{N}_n$, T_p is a map from $\mathcal{X}$ from $\mathcal{X}_0$ (not necessary linear) and $G : \mathcal{X}_0 \times \mathcal{X}_0 \to \mathbb{R}$ is a scalar kernel, then*

$$K(x,t) := \Big(G(T_p x, T_q t)\Big)_{p,q=1}^n, \quad \forall x,t \in \mathcal{X}$$

is a matrix-valued kernel on $\mathcal{X}$.

A specific instance of the above example is described by Vazquez and Walter (2003) in the context of system identification. It corresponds to the choices that $\mathcal{X}_0 = \mathcal{X} = \mathbb{R}$ and $T_p(x) = x + \tau_p$, where $\tau_p \in \mathbb{R}$. In this case, the kernel K models "delays" between the components of the vector-valued function. Indeed, it is easy to verify that, for this choice, for all $f \in \mathcal{H}_K$ and $p \in \mathbb{N}_n$,

$$f_p(x) := (f(x), e_p) = h(x - \tau_p), \quad \forall x \in \mathcal{X}$$

where h is a scalar-valued function in the RKHS of kernel G.

Other choices of the map T_p are possible and provide interesting extensions of scalar kernels. For instance, the choice $K(x,t) := (e^{\sigma_{pq}\langle x,t\rangle} : p,q \in \mathbb{N}_n)$, where $\sigma = (\sigma_{pq})$ is a positive semi-definite matrix suggested by Example 2. Specifically, the eigenvalue decomposition of the matrix σ is given by $\sigma = \sum_{i=1}^n \lambda_i u_i u_i^T$ and, for any $x \in \mathcal{X}$ and $i \in \mathbb{N}_n$ the map $T_p^{(i)}$ is given by $T_p^{(i)} x := \sqrt{\lambda_i} u_{ip} x$. Therefore, we obtain that $K(x,t) = (\prod_{i=1}^n e^{\langle T_p^{(i)} x, T_q^{(i)} t\rangle} : p,q \in \mathbb{N}_n)$ and, so, by Proposition 3, we conclude that K is a matrix-valued kernel.

It is interesting to note, in passing, that, although one would expect the function

$$K(x,t) := \Big(e^{-\sigma_{pq}\|x-t\|^2}\Big)_{p,q=1}^n, \quad \forall x,t \in \mathcal{X} \tag{6}$$

to be a kernel over $\mathcal{X} = \mathbb{R}^d$, we will show later in Section 5 that this is not true, unless all entries of the matrix σ are the same.

Our next example called Hessian of Gaussian is motivated by the problem of learning gradients (Solak et al., 2002; Mukherjee and Zhou, 2006). In many applications, one wants to learn an unknown real-valued function $f(x)$, $x = (x^1, \ldots, x^d) \in \mathbb{R}^d$ and its gradient function $\nabla f = (\partial_1 f, \ldots, \partial_d f)$ where, for any $j \in \mathbb{N}_d$, $\partial_p f$ denotes the p-th partial derivative of f. Here the outputs y_{ip} denotes the observation of derivative of p-th derivative at sample x_i. Therefore, this problem is an appealing example of multi-task learning: learn the target function and its gradient function jointly.

To see why this problem is related with the Hessian of Gaussian, we adopt the Gaussian process (Rasmussen and Williams, 2006) viewpoint of kernel methods. In this perspective, kernels are interpreted as covariance functions of Gaussian prior probability distributions over suitable sets of functions. More specifically, the (unknown) target function f is usually assumed as the realizations of random variables indexed by its input vectors in a zero-mean Gaussian process. The Gaussian process can be fully specified by giving the covariance matrix for any finite set of zero-mean random

variables $\{f(x_i) : i \in \mathbb{N}_m\}$. The covariance between the functions corresponding to the inputs x_i and x_j can be defined by a given Mercer kernel, for example, the Gaussian kernel $G(x) = \exp(-\frac{\|x\|^2}{\sigma})$ with $\sigma > 0$, that is,

$$\operatorname{cov}(f(x_i), f(x_j)) = G(x_i - x_j).$$

Consequently, the covariance between $\partial_p f$ and $\partial_q f$ is given by

$$\operatorname{cov}(\partial_p f(x_i), \partial_q f(x_j)) = \partial_p \partial_q \operatorname{cov}(f(x_i), f(x_j)) = -\partial_p \partial_q G(x_i - x_j).$$

This suggests to us to use the Hessian of Gaussian to model the correlation of gradient function ∇f as we present in the following example.

Example 3 *We let $\mathcal{Y} = \mathcal{X} = \mathbb{R}^n$, and, for any $x = (x_p : p \in \mathbb{N}_n) \in \mathcal{X}$, $G(x) = \exp(-\frac{\|x\|^2}{\sigma})$ with $\sigma > 0$. Then, the Hessian matrix of G given by*

$$K(x,t) := (-(\partial_p \partial_q G)(x-t) : p,q \in \mathbb{N}_n) \ \ \forall x,t \in \mathcal{X}$$

is a matrix-valued kernel.

To illustrate our final example we let $L^2(\mathbb{R})$ be the Hilbert space of square integrable functions on $\mathbb{R}$ with the norm $\|h\|_{L^2}^2 := \int_{\mathbb{R}} h^2(x)dx$. Moreover, we denote by $W^1(\mathbb{R})$ the Sobolev space of order one, which is defined as the space of real-valued functions h on $\mathbb{R}$ whose norm

$$\|h\|_{W^1} := \left(\|h\|_{L^2}^2 + \|h'\|_{L^2}^2\right)^{\frac{1}{2}}$$

is finite.

Example 4 *Let $\mathcal{Y} = L^2(\mathbb{R})$, $\mathcal{X} = \mathbb{R}$ and consider the linear space of functions from $\mathbb{R}$ to $\mathcal{Y}$ which have finite norm*

$$\|f\|^2 = \int_{\mathbb{R}} \left(\|f(x,\cdot)\|_{W^1}^2 + \left\| \frac{\partial f(x,\cdot)}{\partial x} \right\|_{W^1}^2 \right) dx.$$

Then this is an RKHS with multi-task kernel given, for every $x,t \in \mathcal{X}$, by

$$(K(x,t)y)(r) = e^{-\pi|x-t|} \int_{\mathbb{R}} e^{-\pi|r-s|} y(s)ds, \ \ \forall y \in \mathcal{Y}, \ r \in \mathbb{R}.$$

This example may be appropriate to learn the heat distribution in a medium if we think of x as time. Another potential application extends the discussion following Example 1. Specifically, we consider the case that the input x represents both time and a task (e.g., the profile identifying a customer) and the output is the regression function associated to that task (e.g., the preference function of a customer, see Evgeniou et al., 2005, for more information). So, this example may be amenable for learning the dynamics of the tasks.

Further examples for the case that $\mathcal{Y} = L^2(\mathbb{R}^d)$ will be provided in Section 5. We also postpone to that section the proof of the claims in Examples 1-4 as well as the discussion about the universality of the kernels therein.

We end this section with some remarks. It is well known that universality of kernels is a main hypothesis in the proof of the consistency of kernel-based learning algorithms. Universal consistency of learning algorithms and their error analysis also rely on the capacity of the RKHS. In particular,

following the exact procedure for the scalar case in Cucker and Smale (2001), one sufficient condition for universal consistency of vector-valued (multi-task) learning algorithms is the compactness of the unit ball of vector-valued RKHS relative to the space of continuous vector-valued functions. Another alternate sufficient condition was proved in Caponnetto and De Vito (2006) for the regularized least-squares algorithm over vector-valued RKHS. There, it was assumed that, in addition to the universality of the kernel, the trace of the operators $K(x,x)$ is finite, for every $x \in \mathcal{X}$. Clearly, both conditions are fulfilled by the multi-task kernels presented above if the output space $\mathcal{Y}$ is finite dimensional, but they become non trivial in the infinite dimensional case. However, it is not clear to the authors whether either of these two conditions is necessary for universal consistency. We hope to come back to this problem in the future.

3. Universal Kernels by Features

In this section, we prove that a multi-task kernel is universal if and only if *its feature representation is universal*. To explain what we have in mind, we require some additional notation. We let $\mathcal{W}$ be a Hilbert space and $\mathcal{L}(\mathcal{Y},\mathcal{W})$ be the set of all bounded linear operators from $\mathcal{Y}$ to $\mathcal{W}$. A *feature representation* associated with a multi-task kernel K is a continuous function

$$\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{Y},\mathcal{W})$$

such that, for every $x,t \in \mathcal{X}$

$$K(x,t) = \Phi^*(x)\Phi(t), \tag{7}$$

where, we recall, for each $x \in \mathcal{X}$, $\Phi^*(x)$ is the adjoint of $\Phi(x)$ and, therefore, it is in $\mathcal{L}(\mathcal{W},\mathcal{Y})$. Hence, from now on we call $\mathcal{W}$ the *feature space*. In the case that $\mathcal{Y} = \mathbb{R}$, the condition that $\Phi(x) \in \mathcal{L}(\mathcal{Y},\mathcal{W})$ can be merely viewed as saying that $\Phi(x)$ in an element of $\mathcal{W}$. Therefore, at least in this case we can rewrite Equation (7) as

$$K(x,t) = (\Phi(x),\Phi(t))_{\mathcal{W}}. \tag{8}$$

Another example of practical importance corresponds to the choice $\mathcal{W} = \mathbb{R}^k$ and $\mathcal{Y} = \mathbb{R}^n$, both finite dimensional Euclidean spaces. Here we can identify $\Phi(x)$ relative to standard basis of $\mathcal{W}$ and $\mathcal{Y}$ with the $k \times n$ matrix $\Phi(x) = (\Phi_{rp}(x) : r \in \mathbb{N}_k, p \in \mathbb{N}_n)$, where Φ_{rp} are scalar-valued continuous functions on $\mathcal{X}$. Therefore, according to (7) the matrix representation of the multi-task kernel K is, for each $x,t \in \mathcal{X}$,

$$(K(x,t))_{pq} = \sum_{r \in \mathbb{N}_k} \Phi_{rp}(x)\Phi_{rq}(t), \quad p,q \in \mathbb{N}_n.$$

Returning to the general case, we emphasize that we *assume* that the kernel K has the representation in Equation (7), although if it corresponds to a compact integral operator, such a representation will follow from the spectral theorem and Mercer Theorem (see, e.g., Micchelli et al., 2006).

Associated with a feature representation as described above is the following closed linear subspace of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$

$$\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y}) := \overline{\{\Phi^*(\cdot)w : w \in \mathcal{W}\}}, \tag{9}$$

where the closure is taken relative to the norm of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$. The continuity of the functions $\Phi^*(\cdot)w$ follows from the assumed continuity of $K(\cdot,\cdot)$ by

$$\begin{aligned} \|\Phi^*(x)w - \Phi^*(t)w\|^2 &\le \|\Phi^*(x) - \Phi^*(t)\|^2 \|w\|^2_{\mathcal{W}} \\ &= \|(\Phi^*(x) - \Phi^*(t))(\Phi(x) - \Phi(t))\| \|w\|^2_{\mathcal{W}} \\ &= \|K(x,x) + K(t,t) - K(x,t) - K(t,x)\| \|w\|^2_{\mathcal{W}}. \end{aligned}$$

Our definition of the phrase "the feature representation is universal" means that $\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y}) = \mathcal{C}(\mathcal{Z},\mathcal{Y})$ for every compact subset $\mathcal{Z}$ of the input space $\mathcal{X}$. The theorem below demonstrates, as we mentioned above, that the kernel K is universal if and only if its feature representation is universal. The content of Theorem 4 and of the other results of this Section (Lemmas 5, 6 and Proposition 7) are graphically represented by the diagram in Table 2

Theorem 4 *If K is a continuous multi-task kernel with feature representation Φ, then for every compact subset $\mathcal{Z}$ of $\mathcal{X}$, we have that $\mathcal{C}_K(\mathcal{Z},\mathcal{Y}) = \mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$.*

Proof The theorem follows straightforwardly from Lemmas 5, 6 and Proposition 7, which we present below.

■

As we know, the feature representation of a given kernel is not unique, therefore we conclude by Theorem 4 that if *some* feature representation of a multi-task kernel is universal then *every* feature representation is universal.

We shall give two different proofs of this general theorem. The first one will use a technique highlighted in Micchelli and Pontil (2005) and will be given in this section. The second proof will be given in the next section and uses the notion of *vector measure*. Both approaches adopt the point of view of Micchelli et al. (2006), in which Theorem 4 is proved in the special case that $\mathcal{Y} = \mathbb{R}$.

We now begin to explain in detail our first proof. We denote the unit ball in $\mathcal{Y}$ by $\mathcal{B}_1 := \{y : y \in \mathcal{Y}, \|y\| \leq 1\}$ and let $\mathcal{Z}$ be a prescribed compact subset of $\mathcal{X}$. Recall that $\mathcal{B}_1$ is not compact in the norm topology on $\mathcal{Y}$ *unless* $\mathcal{Y}$ is finite dimensional. But it is compact in the *weak topology* on $\mathcal{Y}$ since $\mathcal{Y}$ is a Hilbert space (see, e.g., Yosida, 1980). Remember that a basis for the open neighborhood of the origin in the weak topology is a set of the form $\{y : y \in \mathcal{Y}, |(y,y_i)| \leq 1, i \in \mathbb{N}_m\}$, where $y_1,\ldots,y_m$ are arbitrary vectors in $\mathcal{Y}$. We put on $\mathcal{B}_1$ the weak topology and conclude, by Tychonoff's theorem (see, e.g., Folland, 1999, p.136), that the set $\mathcal{Z}\times\mathcal{B}_1$ is also compact in the product topology.

The above observation allows us to associate $\mathcal{Y}$-valued functions defined on $\mathcal{Z}$ to scalar-valued functions defined on $\mathcal{Z}\times\mathcal{B}_1$. Specifically, we introduce the map $\iota : \mathcal{C}(\mathcal{Z},\mathcal{Y}) \to \mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$ which maps any function $f \in \mathcal{C}(\mathcal{Z},\mathcal{Y})$ to the function $\iota(f) \in \mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$ defined by the action

$$\iota(f) : (x,y) \mapsto (f(x),y)_\mathcal{Y}, \quad \forall (x,y) \in (\mathcal{Z}\times\mathcal{B}_1). \tag{10}$$

Consequently, it follows that the map ι is *isometric*, since

$$\sup_{x\in\mathcal{Z}} \|f(x)\|_\mathcal{Y} = \sup_{x\in\mathcal{Z}} \sup_{\|y\|\leq 1} |(f(x),y)_\mathcal{Y}| = \sup_{x\in\mathcal{Z}} \sup_{y\in\mathcal{B}_1} |\iota(f)(x,y)|,$$

where the first equality follows by Cauchy-Schwarz inequality. Moreover, we will denote by $\iota(\mathcal{C}(\mathcal{Z},\mathcal{Y}))$ the image of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ under the mapping ι. In particular, this space is a closed linear subspace of $\mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$ and, hence, a Banach space.

Similarly, to any multi-task kernel K on $\mathcal{Z}$ we associate a scalar kernel J on $\mathcal{Z}\times\mathcal{B}_1$ defined, for every $(x,y),(t,u) \in \mathcal{X}\times\mathcal{B}_1$, as

$$J((x,y),(t,u)) := (K(x,t)u,y). \tag{11}$$

Moreover, we denote by $\mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$ the closure in $\mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$ of the set of the sections of the kernel, $\{J((x,y),(\cdot,\cdot)) : (x,y) \in \mathcal{Z}\times\mathcal{B}_1\}$. It is important to realize that whenever K is a valid multi-task kernel, then J is a valid scalar kernel.

$$\begin{array}{ccc} \mathcal{C}_\Psi(\mathcal{Z}\times\mathcal{B}_1) & = & \mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1) \\ \iota\uparrow & & \uparrow\iota \\ \mathcal{C}_K(\mathcal{Z},\mathcal{Y}) & = & \mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y}) \end{array}$$

Table 2: The top equality is Proposition 7, the bottom equality is Theorem 4 and the left and right arrows are Lemma 5 and 6, respectively.

The lemma below relates the set $\mathcal{C}_K(\mathcal{Z},\mathcal{Y})$ to the corresponding set $\mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$ for the kernel J on $\mathcal{Z}\times\mathcal{B}_1$.

Lemma 5 *If $\mathcal{Z}$ is a compact subset of $\mathcal{X}$ and K is a continuous multi-task kernel then $\iota(\mathcal{C}_K(\mathcal{Z},\mathcal{Y})) = \mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$.*

Proof The assertion follows by Equation (11) and the continuity of the map ι. ■

In order to prove Theorem 4, we also need to provide a similar lemma for the set $\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$. Before we state the lemma, we note that knowing the features of the multi-task kernel K leads us to the features for the scalar-kernel J associated to K. Specifically, for every $(x,y),(t,u)\in\mathcal{X}\times\mathcal{B}_1$, we have that

$$J((x,y),(t,u)) = (\Psi(x,y),\Psi(t,u))_{\mathcal{W}}, \tag{12}$$

where the continuous function $\Psi : \mathcal{X}\times\mathcal{B}_1 \to \mathcal{W}$ is defined as

$$\Psi(x,y) = \Phi(x)y, \quad x\in\mathcal{X}, y\in\mathcal{B}_1.$$

Thus, Equation (12) parallels Equation (8) except that $\mathcal{X}$ is replaced by $\mathcal{X}\times\mathcal{B}_1$. We also denote by $\mathcal{C}_\Psi(\mathcal{Z}\times\mathcal{B}_1) = \overline{\{(\Psi(\cdot),w)_{\mathcal{W}} : w\in\mathcal{W}\}}$, the closed linear subspace of $\mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$.

Lemma 6 *If $\mathcal{Z}$ is a compact subset of $\mathcal{X}$ and K is a continuous multi-task kernel with feature representation Φ then $\iota(\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})) = \mathcal{C}_\Psi(\mathcal{Z}\times\mathcal{B}_1)$.*

Proof The proof is immediate. Indeed, for each $x\in\mathcal{X}$, $w\in\mathcal{W}$, $y\in\mathcal{Y}$, we have that $(\Phi^*(x)w,y)_{\mathcal{Y}} = (w,\Phi(x)y)_{\mathcal{W}} = (\Psi(x,y),w)_{\mathcal{W}}$. ■

To complete the proof of Theorem 4, as illustrated in Table 2 it suffices to show that $\mathcal{C}_\Psi(\mathcal{Z}\times\mathcal{B}_1) = \mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$. To this end, we review some facts about signed measures and bounded linear functionals on continuous functions. Let Ω be any prescribed *compact* Hausdorff space and $\mathcal{C}(\Omega)$ be the space of all real-valued continuous functions with norm $\|\cdot\|_{\infty,\Omega}$. We also use the notation $\mathscr{B}(\Omega)$ to denote the Borel σ-algebra on Ω. Now, we recall the description of the dual space of $\mathcal{C}(\Omega)$. By the Riesz representation theorem, any linear functional L in the dual space of $\mathcal{C}(\Omega)$ is uniquely identified as a regular signed Borel measure ν on Ω (see, e.g., Folland 1999), that is,

$$L(g) = \int_\Omega g(x)d\nu(x), \quad \forall g\in\mathcal{C}(\Omega).$$

The variation of ν is given, for any $E \in \mathscr{B}(\Omega)$, by

$$|\nu|(E) := \sup\Big\{ \sum_{j\in\mathbb{N}} |\nu(A_j)| : \{A_j : j \in \mathbb{N}\} \text{ pairwise disjoint and } \cup_{j\in\mathbb{N}} A_j = E \Big\}.$$

Moreover, we have that $\|L\| = \|\nu\|$, where $\|\nu\| = |\nu|(\Omega)$ and $\|L\|$ is the operator norm of L defined by $\|L\| = \sup_{\|g\|_{\infty,\Omega}=1} |L(g)|$. Recall that a Borel measure ν is *regular* if, for any $E \in \mathscr{B}(\mathcal{X})$,

$$\nu(E) = \inf\big\{\nu(U) : E \subseteq U, U \text{ open}\big\} = \sup\big\{\nu(\bar{U}) : \bar{U} \subseteq E, \bar{U} \text{ compact}\big\}.$$

In particular, every finite Borel measure on Ω is regular, see Folland (1999, p.217). We denote by $\mathcal{M}(\Omega)$ the space of all regular signed measures on Ω with total variation norm. We emphasize here that the Riesz representation theorem stated above requires the compactness of the underlying space Ω.

As mentioned above, $\mathcal{Z} \times \mathcal{B}_1$ is compact relative to the weak topology if $\mathcal{Z}$ is compact. This enables us to use the Riesz representation theorem on the underlying space $\Omega = \mathcal{Z} \times \mathcal{B}_1$ to show the following proposition.

Proposition 7 *For any compact set $\mathcal{Z} \subseteq \mathcal{X}$, and any continuous multi-task kernel K with feature representation Φ, we have that $\mathcal{C}_\Psi(\mathcal{Z} \times \mathcal{B}_1) = \mathcal{C}_J(\mathcal{Z} \times \mathcal{B}_1)$.*

Proof For any compact set $\mathcal{Z} \subseteq \mathcal{X}$, recall that $\mathcal{Z} \times \mathcal{B}_1$ is compact if $\mathcal{B}_1$ is endowed with the weak topology of $\mathcal{Y}$. Hence, the result follows by applying Theorem 4 in Micchelli et al. (2006) to the scalar kernel J on the set $\mathcal{Z} \times \mathcal{B}_1$ with the feature representation given by Equation (12). However, for the convenience of the reader we review the steps of the argument used to prove that theorem. The basic idea is the observation that two closed subspaces of a Banach space are equal if and only if whenever a continuous linear functional vanishes on either one of the subspaces, it must also vanish on the other one. This is a consequence of the Hahn-Banach Theorem (see, e.g., Lax, 2002). In the case at hand, we know by the Riesz Representation Theorem that all continuous linear functionals L on $\mathcal{C}(\mathcal{Z} \times \mathcal{B}_1)$ are given by a regular signed Borel measure ν, that is for every $F \in \mathcal{C}(\mathcal{Z} \times \mathcal{B}_1)$, we have that

$$L(F) = \int_{\mathcal{Z}\times\mathcal{B}_1} F(x,y) d\nu(x,y).$$

Now, suppose that L vanishes on $\mathcal{C}_J(\mathcal{Z} \times \mathcal{B}_1)$, then we conclude, by (12), that

$$0 = \int_{\mathcal{Z}\times\mathcal{B}_1} \int_{\mathcal{Z}\times\mathcal{B}_1} (\Psi(x,y), \Psi(t,u))_{\mathcal{W}} d\nu(x,y) d\nu(t,u).$$

Also, since K is assumed to be continuous relative to the operator norm and $\mathcal{Z}$ is compact we have that $\|\Psi(x,y)\|_{\mathcal{W}}^2 = \|\Psi(x)y\|_{\mathcal{W}}^2 = (K(x,x)y,y) \leq \sup_{x\in\mathcal{Z}} \|K(x,x)\| < \infty$. This together with the equation

$$\left\| \int_{\mathcal{Z}\times\mathcal{B}_1} \Psi(x,y) d\nu(x,y) \right\|_{\mathcal{W}} \leq \int_{\mathcal{Z}\times\mathcal{B}_1} \|\Psi(x,y) d\nu(x,y)\| \, d\nu(x,y) \leq \sup_x \|K(x,x)\| |\nu|(\mathcal{Z}\times\mathcal{B}_1)$$

imply that the integrand $\int_{\mathcal{Z}\times\mathcal{B}_1} \Psi(x,y) d\nu(x,y)$ exists. Consequently, it follows that

$$\int_{\mathcal{Z}\times\mathcal{B}_1} \int_{\mathcal{Z}\times\mathcal{B}_1} (\Psi(x,y), \Psi(t,u))_{\mathcal{W}} d\nu(x,y) d\nu(t,u) = \left\| \int_{\mathcal{Z}\times\mathcal{B}_1} \Psi(x,y) d\nu(x,y) \right\|_{\mathcal{W}}^2 \tag{13}$$

and, so, we conclude that

$$\int_{\mathcal{Z}\times\mathcal{B}_1} \Psi(x,y)d\nu(x,y) = 0. \tag{14}$$

The proof of Equation (13) and the interpretation of the $\mathcal{W}$-valued integral appearing in Equation (14) is explained in detail in Micchelli et al. (2006). So, we conclude that L vanishes on $\mathcal{C}_\Psi(\mathcal{Z}\times\mathcal{B}_1)$.

Conversely, if L vanishes on $\mathcal{C}_\Psi(\mathcal{Z}\times\mathcal{B}_1)$ then, for any $x\in\mathcal{Z}, y\in\mathcal{B}_1$, we have that

$$\int_{\mathcal{Z}\times\mathcal{B}_1} J((x,y),(t,u))d\nu(t,u) = \left(\Psi(x,y), \int_{\mathcal{Z}\times\mathcal{B}_1} \Psi(t,u)\nu(t,u)\right) = 0$$

that is, L vanishes on $\mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$. ■

4. Further Perspectives for Universality

In this section, we provide an alternate proof of Theorem 4 using the notion of vector measure and also highlight the notion of the annihilator of a set, a useful tool for our examples of multi-task kernels in Section 5.

At first glance, the reduction of the question of when a multi-task kernel is universal to the scalar case, as explained in Section 3, seems compelling. However, there are several reasons to explore alternate approaches to this problem. Firstly, from a practical point of view, if we view multi-task learning as a scalar problem we may loose the ability to understand cross task interactions. Secondly, only one tool to resolve a problem may limit the possibility of success. Finally, as we demonstrated in Section 3 universality of multi-task kernels concerns density in the subspace $\mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$, not the full space $\mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$, an issue heretofore not considered. Therefore, we cannot directly employ the methods of the scalar case as presented in (Micchelli et al., 2003) to the multi-task case.

As we shall see in this section, the concept of vector measure allows us to directly confront the set $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ rather than following a circuitous path to $\mathcal{C}_J(\mathcal{Z}\times\mathcal{B}_1)$. However, the basic principle which we employ is the same, namely, two closed linear subspaces of a Banach space are equal if and only if whenever a bounded linear functional vanishes on one of them, it also vanishes on the other one. Thus, to implement this principle we are led to consider the dual space of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$. We remark, in passing, that this space also arose in the context of the feature space perspective for learning the kernel, see Micchelli and Pontil (2005a). For a description of the dual space of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$, we need the notion of vector measures and in this regard rely upon the information about them in Diestel and Uhl, Jr. (1977).

To introduce our first definition, recall that throughout this paper $\mathcal{X}$ denotes a Hausdorff space, $\mathcal{Z}\subseteq\mathcal{X}$ any compact subset of $\mathcal{X}$ and $\mathscr{B}(\mathcal{Z})$ the Borel σ-algebra of $\mathcal{Z}$.

Definition 8 *A map $\mu:\mathscr{B}(\mathcal{Z})\to\mathcal{Y}$ is called a Borel vector measure if μ is countably additive, that is, $\mu(\cup_{j=1}^\infty E_j) = \sum_{j=1}^\infty \mu(E_j)$ in the norm of $\mathcal{Y}$, for all sequences $\{E_j : j\in\mathbb{N}\}$ of pairwise disjoint sets in $\mathscr{B}(\mathcal{Z})$*

It is important to note that the definition of vector measure given in Diestel and Uhl, Jr. (1977) only requires it to be finitely additive. For our purpose here, we only use countably additive measures and thus do not require the more general setting used in Diestel and Uhl, Jr. (1977).

For any vector measure μ, the variation of μ is defined, for any $E \in \mathscr{B}(\mathcal{Z})$, by the equation

$$|\mu|(E) := \sup\left\{ \sum_{j\in\mathbb{N}} \|\mu(A_j)\| : \{A_j : j \in \mathbb{N}\} \text{ pairwise disjoint and } \cup_{j\in\mathbb{N}} A_j = E \right\}.$$

In our terminology we conclude from (Diestel and Uhl, Jr., 1977, p.3) that μ is a vector measure if and only if the corresponding variation $|\mu|$ is a scalar measure as explained in Section 3. Whenever $|\mu|(\mathcal{Z}) < \infty$, we call μ a vector measure of *bounded variation* on $\mathcal{Z}$. Moreover, we say that a Borel vector measure μ on $\mathcal{Z}$ is *regular* if its variation measure $|\mu|$ is regular as defined in Section 3. We denote by $\mathcal{M}(\mathcal{Z},\mathcal{Y})$ the Banach space of all vector measures with bounded variation and norm $\|\mu\| := |\mu|(\mathcal{Z})$.

For any scalar measure $\nu \in \mathcal{M}(\mathcal{Z}\times\mathcal{B}_1)$, we define a $\mathcal{Y}$-valued function on $\mathscr{B}(\mathcal{Z})$, by the equation

$$\mu(E) := \int_{E\times\mathcal{B}_1} y d\nu(x,y), \qquad \forall E \in \mathscr{B}(\mathcal{Z}). \tag{15}$$

Let us confirm that μ is indeed a vector measure. For this purpose, choose any sequence of pairwise disjoint subsets $\{E_j : j \in \mathbb{N}\} \subseteq \mathscr{B}(\mathcal{Z})$, and observe that

$$\sum_{j\in\mathbb{N}} \|\mu(E_j)\|_{\mathcal{Y}} \leq \sum_{j\in\mathbb{N}} \Big| \int_{E_j}\int_{\mathcal{B}_1} d\nu(x,y)\Big| \leq |\nu|(\mathcal{Z}\times\mathcal{B}_1),$$

which implies that $|\mu|(\mathcal{Z})$ is finite and, hence, μ is a regular vector measure. This observation suggests that we define, for any $f \in \mathcal{C}(\mathcal{Z},\mathcal{Y})$, the integral of f relative to μ as

$$\int_{\mathcal{Z}} (f(x), d\mu(x)) := \int_{\mathcal{Z}}\int_{\mathcal{B}_1} (f(x), y) d\nu(x,y). \tag{16}$$

Alternatively, by the standard techniques of measure theory, for any vector measure $\mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y})$ the integral $\int_{\mathcal{Z}} (f(x), d\mu(x))$ is well-defined.

One of our goals below is to show that given any vector measure μ, there corresponds a scalar measure ν such that Equation (16) still holds. Before doing so, let us point out a useful property about the integral appearing in the left hand side of Equation (16). Specifically, for any $y \in \mathcal{Y}$, we associate to any $\mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y})$, a scalar measure μ_y defined, for any $E \in \mathscr{B}(\mathcal{Z})$, by the equation $\mu_y(E) := (y, \mu(E))$. Therefore, we conclude, for any $f \in \mathcal{C}(\mathcal{Z})$, that

$$\int_{\mathcal{Z}} (y f(x), d\mu(x)) = \int_{\mathcal{Z}} f(x) d\mu_y(x).$$

To prepare for our description of the dual space of $\mathcal{C}(\mathcal{Z},\mathcal{Y})$, we introduce, for each $f \in \mathcal{C}(\mathcal{Z},\mathcal{Y})$, a linear functional L_μ defined by,

$$L_\mu f := \int_{\mathcal{Z}} (f(x), d\mu(x)). \tag{17}$$

Then, we have the following useful lemmas, see the appendix for their proofs.

Lemma 9 *If $\mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y})$ then $L_\mu \in \mathcal{C}^*(\mathcal{Z},\mathcal{Y})$ and $\|L_\mu\| = \|\mu\|$.*

Lemma 10 *(Dinculeanu-Singer) For any compact set $\mathcal{Z} \subseteq \mathcal{X}$, the map $\mu \mapsto L_\mu$ is an isomorphism from $\mathcal{M}(\mathcal{Z},\mathcal{Y})$ to $C^*(\mathcal{Z},\mathcal{Y})$.*

Lemma 10 is a vector-valued form of the Riesz representation theorem called *Dinculeanu-Singer theorem*, (see, e.g., Diestel and Uhl, Jr., 1977, p.182). For completeness, we provide a self-contained proof in the appendix.

It is interesting to remark that, for any $\mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y})$ we have established in the proof of Lemma 10 that there exists a regular scalar measure ν on $\mathcal{Z} \times \mathcal{B}_1$ such that

$$L_\mu f = \int_{\mathcal{Z}\times\mathcal{B}_1} (f(x),y) d\nu(x,y).$$

Since we established the isometry between $\mathcal{C}^*(\mathcal{Z},\mathcal{Y})$ and $\mathcal{M}(\mathcal{Z},\mathcal{Y})$, it follows that, for every regular vector measure there corresponds a scalar measure on $\mathcal{Z} \times \mathcal{B}_1$ for which Equation (15) holds true.

In order to provide our alternate proof of Theorem 4, we need to attend to one further issue. Specifically, we need to define the integral $\int_{\mathcal{Z}} K(t,x)(d\mu(x))$ as an element in $\mathcal{Y}$. For this purpose, for any $\mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y})$ and $t \in \mathcal{Z}$ we define a linear functional L_t on $\mathcal{Y}$ at $y \in \mathcal{Y}$ as

$$L_t y := \int_{\mathcal{Z}} (K(x,t)y, d\mu(x)).$$

Since its norm has the property $\|L_t\| \leq \big(\sup_{x\in\mathcal{Z}} \|K(x,t)\|\big)\|\mu\|$, by the Riesz representation lemma, we conclude that there exists a unique element $\bar{y}$ in $\mathcal{Y}$ such that

$$\int_{\mathcal{Z}} (K(x,t)y, d\mu(x)) = (\bar{y}, y).$$

It is this vector $\bar{y}$ which we denote by the integral $\int_{\mathcal{Z}} K(t,x)(d\mu(x))$.

Similarly, we define the integral $\int_{\mathcal{Z}} \Phi(x)(d\mu(x))$ as an element in $\mathcal{W}$. To do this, we note that $\|\Phi(x)\| = \|\Phi^*(x)\|$ and $\|\Phi^*(x)y\|^2 = \langle K(x,x)y, y\rangle$. Hence, we conclude that there exists a constant κ such that, for all $x \in \mathcal{X}$, $\|\Phi(x)\| \leq \|K(x,x)\|^{\frac{1}{2}} \leq \kappa$. Consequently, the linear functional L on $\mathcal{W}$ defined, for any $w \in \mathcal{W}$, by

$$L(w) := \int_{\mathcal{Z}} \Big(\Phi^*(x)w, d\mu(x)\Big)$$

satisfies the inequality $\|L\| \leq \kappa\|\mu\|$. Hence, we conclude that there exists a unique element $\bar{w} \in \mathcal{W}$ such that $L(w) = (\bar{w}, w)$ for any $w \in \mathcal{W}$. Now, we denote $\bar{w}$ by $\int_{\mathcal{Z}} \Phi(x)(d\mu(x))$ which means that

$$\Big(\int_{\mathcal{Z}} \Phi(x)(d\mu(x)), w\Big)_{\mathcal{W}} = \int_{\mathcal{Z}} \Big(\Phi^*(x)w, d\mu(x)\Big). \tag{18}$$

We have now assembled all the necessary properties of vector measures to provide an alternate proof of Theorem 4.

Alternate Proof of Theorem 4. We see from the feature representation (7) that

$$\int_{\mathcal{Z}} K(t,x)(d\mu(x)) = \int_{\mathcal{Z}} \Phi^*(t)\Phi(x)(d\mu(x)) = \Phi^*(t)\Big(\int_{\mathcal{Z}} \Phi(x)(d\mu(x))\Big), \quad \forall t \in \mathcal{Z}.$$

From this equation, we easily see that if $\int_{\mathcal{Z}} \Phi(x)(d\mu(x)) = 0$ then, for every $t \in \mathcal{Z}$, $\int_{\mathcal{Z}} K(t,x)(d\mu(x)) = 0$. On the other hand, applying (18) with the choice $w = \int_{\mathcal{Z}} \Phi(x)(d\mu(x))$ we get

$$\int_{\mathcal{Z}} \left(\Phi^*(t) \int_{\mathcal{Z}} \Phi(x)(d\mu(x)), d\mu(t) \right) = \left\| \int_{\mathcal{Z}} \Phi(x)(d\mu(x)) \right\|_{\mathcal{W}}^2.$$

Therefore, if, for any $t \in \mathcal{Z}$, $\int_{\mathcal{Z}} K(t,x)(d\mu(x)) = 0$ then $\int_{\mathcal{Z}} \Phi(x)(d\mu(x)) = 0$, or equivalently, by Equation (18),

$$\int_{\mathcal{Z}} (\Phi^*(x)w, d\mu(x)) = 0, \quad \forall w \in \mathcal{W}.$$

Consequently, a linear functional vanishes on $\mathcal{C}_K(\mathcal{Z},\mathcal{Y})$ if and only if it vanishes on $\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$ and thus, we obtained that $\mathcal{C}_K(\mathcal{Z},\mathcal{Y}) = \mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$. ■

We end this section with a review of our approach to the question of universality of multi-task kernels. The principal tool we employ is a notion of functional analysis referred to as the *annihilator* set. Recall the notion of the annihilator of a set $\mathcal{V}$ which is defined by

$$\mathcal{V}^\perp := \left\{ \mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y}) \ : \ \int_{\mathcal{Z}} (v(x), d\mu(x)) = 0, \forall v \in \mathcal{V} \right\}.$$

Notice that the annihilator of the closed linear span of $\mathcal{V}$ is the same as that of $\mathcal{V}$. Consequently, by applying the basic principle stated at the beginning of this section , we conclude that the linear span of $\mathcal{V}$ is dense in $\mathcal{C}(\mathcal{Z},\mathcal{Y})$, that is, $\overline{\text{span}}(\mathcal{V}) = \mathcal{C}(\mathcal{Z},\mathcal{Y})$ if and only if the annihilator $\mathcal{V}^\perp = \{0\}$. Hence, applying this observation to the set of kernel sections $K(\mathcal{Z}) := \{K(\cdot,x)y : \ x \in \mathcal{Z}, \ y \in \mathcal{Y}\}$ or to the set of its corresponding feature sections $\Phi(\mathcal{Z}) := \{\Phi^*(\cdot)w : w \in \mathcal{W}\}$, we obtain from Lemma 10 and Theorem 4, the summary of our main result.

Theorem 11 *Let $\mathcal{Z}$ be a compact subset of $\mathcal{X}$, K a continuous multi-task kernel, and Φ its feature representation. Then, the following statements are equivalent.*

1. $\mathcal{C}_K(\mathcal{Z},\mathcal{Y}) = \mathcal{C}(\mathcal{Z},\mathcal{Y})$.
2. $\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y}) = \mathcal{C}(\mathcal{Z},\mathcal{Y})$.
3. $K(\mathcal{Z})^\perp = \left\{ \mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y}) \ : \ \int_{\mathcal{Z}} K(t,x)(d\mu(x)) = 0, \ \forall t \in \mathcal{Z} \right\} = \{0\}$.
4. $\Phi(\mathcal{Z})^\perp = \left\{ \mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y}) \ : \ \int_{\mathcal{Z}} \Phi(x)(d\mu(x)) = 0 \right\} = \{0\}$.

5. Universal Kernels

In this section, we prove the universality of some kernels, based on Theorem 11 developed above. Specifically, the examples highlighted in Section 2 will be discussed in detail.

Kernel's universality is a main hypothesis in the proof of consistency of learning algorithms. Universal consistency of the regularized least-squares algorithm over vector-valued RKHS was proved in Caponnetto and De Vito (2006); there, it was assumed that, in addition to universality of the kernel, the trace of the operators $K(x,x)$ is finite. In particular, this extra condition on the kernel holds, for the Example 1 highlighted in Section 2, when the operators B_j are trace class, and does not hold for Example 4. It is not clear to the authors whether the finite trace condition is necessary for consistency.

5.1 Product of Scalar Kernels and Operators

Our first example is produced by coupling a scalar kernel with an operator in $\mathcal{L}_+(\mathcal{Y})$. Given a scalar kernel G on $\mathcal{X}$ and an operator $B \in \mathcal{L}_+(\mathcal{Y})$, we define the function $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$ by

$$K(x,t) = G(x,t)B, \quad \forall x,t \in \mathcal{X}. \tag{19}$$

For any $\{x_j \in \mathcal{X} : j \in \mathbb{N}_m\}$ and $\{y_j \in \mathcal{Y} : j \in \mathbb{N}_m\}$, we know that $(G(x_i,x_j))_{i,j\in\mathbb{N}_m}$ and $((By_i,y_j))_{i,j\in\mathbb{N}_m}$ are positive semi-definite. Applying Schur's lemma implies that the matrix $(G(x_i,x_j)(By_i,y_j))_{i,j\in\mathbb{N}_m}$ is positive semi-definite and hence, K is positive semi-definite. Moreover, $K^*(x,t) = K(x,t) = K(t,x)$ for any $x,t \in \mathcal{X}$. Therefore, we conclude by Definition 1 that K is a multi-task kernel.

Our goal below is to use the feature representation of the scalar kernel G to introduce the corresponding one for kernel K. To this end, we first let $\mathcal{W}$ be a Hilbert space and $\phi : \mathcal{X} \to \mathcal{W}$ a feature map of the scalar kernel G, so that

$$G(x,t) = (\phi(x),\phi(t))_{\mathcal{W}}, \qquad \forall x,t \in \mathcal{X}.$$

Then, we introduce the *tensor vector space* $\mathcal{W} \bigotimes \mathcal{Y}$. Algebraically, this vector space is spanned by elements of the form $w \otimes y$ with $w \in \mathcal{W}, y \in \mathcal{Y}$. For any $w_1 \otimes y_1, w_2 \otimes y_2 \in \mathcal{W} \bigotimes \mathcal{Y}$ and $c \in \mathbb{R}$, there holds the multi-linear relation

$$cw \otimes y = w \otimes cy = c(w \otimes y), \quad (w_1 + w_2) \otimes y = w_1 \otimes y + w_2 \otimes y,$$

and

$$w \otimes (y_1 + y_2) = w \otimes y_1 + w \otimes y_2.$$

We can turn the tensor space into an inner product space by defining, for any $w_1 \otimes y_1, w_2 \otimes y_2 \in \mathcal{W} \bigotimes \mathcal{Y}$,

$$\langle w_1 \otimes y_1, w_2 \otimes y_2 \rangle = (w_1,w_2)_{\mathcal{W}}(y_1,y_2)_{\mathcal{Y}}$$

and extending by linearity. Finally, taking the completion under this inner product, the vector space $\mathcal{W} \bigotimes \mathcal{Y}$ becomes a Hilbert space. Furthermore, if $\mathcal{W}$ and $\mathcal{Y}$ have orthonormal bases $\{w_i : i \in \mathbb{N}\}$ and $\{y_i : i \in \mathbb{N}\}$ respectively, then $\mathcal{W} \bigotimes \mathcal{Y}$ is exactly the Hilbert space spanned by the orthonormal basis $\{w_i \otimes y_j : i,j \in \mathbb{N}\}$ under the inner product defined above. For instance, if $\mathcal{W} = \mathbb{R}^k$ and $\mathcal{Y} = \mathbb{R}^n$, then $\mathcal{W} \bigotimes \mathcal{Y} = \mathbb{R}^{kn}$.

The above tensor product suggests that we define the map $\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{W} \otimes \mathcal{Y})$ of kernel K by

$$\Phi(x)y := \phi(x) \otimes \sqrt{B}y, \qquad \forall x \in \mathcal{X},\ y \in \mathcal{Y},$$

and it follows that $\Phi^* : \mathcal{X} \to \mathcal{L}(\mathcal{W} \otimes \mathcal{Y}, \mathcal{Y})$ is given by

$$\Phi^*(x)(w \otimes y) := (\phi(x),w)_{\mathcal{W}}\sqrt{B}y, \quad \forall x \in \mathcal{X}, w \in \mathcal{W}, \text{ and } y \in \mathcal{Y}. \tag{20}$$

From the above observation, it is easy to check, for any $x,t \in \mathcal{X}$ and $y,u \in \mathcal{Y}$, that $(K(x,t)y,u) = \langle \Phi(x)y, \Phi(t)u \rangle$. Therefore, we conclude that Φ is a feature map for the multi-task kernel K.

Finally, we say that an operator $B \in \mathcal{L}_+(\mathcal{Y})$ is *positive definite* if (By,y) is positive whenever y is nonzero. We are now ready to present the result on universality of kernel K.

Theorem 12 *Let $G : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a continuous scalar kernel, $B \in \mathcal{L}_+(\mathcal{Y})$ and K be defined by Equation (19). Then, K is a multi-task universal kernel if and only if the scalar kernel G is universal and the operator B is positive definite.*

Proof By Theorem 11 and the feature representation (20), we only need to show that $\Phi(\mathcal{Z})^\perp = \{0\}$ if and only if G is universal and the operator B is positive definite.

We begin with the sufficiency. Suppose that there exists a nonzero vector measure μ such that, for any $w \otimes y \in \mathcal{W} \otimes \mathcal{Y}$, there holds

$$\int_{\mathcal{Z}} (\Phi^*(x)(w \otimes y), d\mu(x)) = \int_{\mathcal{Z}} (\phi(x), w)_{\mathcal{W}} (\sqrt{B}y, d\mu(x)) = 0. \tag{21}$$

Here, with a little abuse of notation we interpret, for a fixed $y \in \mathcal{Y}$, $(\sqrt{B}y, d\mu(x))$ as a scalar measure defined, for any $E \in \mathscr{B}(\mathcal{Z})$, by

$$\int_E (\sqrt{B}y, d\mu(x)) = (\sqrt{B}y, \mu(E)).$$

Since $\mu \in \mathcal{M}(\mathcal{Z}, \mathcal{Y})$, $(\sqrt{B}y, d\mu(x))$ is a regular signed scalar measure. Therefore, we see from (21) that $(\sqrt{B}y, d\mu(x)) \in \phi(\mathcal{Z})^\perp$ for any $y \in \mathcal{Y}$. Remember that G is universal if and only if $\phi(\mathcal{Z})^\perp = \{0\}$, and thus we conclude from (21) that $(\sqrt{B}y, d\mu(x)) = 0$ for any $y \in \mathcal{Y}$. It follows that $(\sqrt{B}y, \mu(E)) = 0$ for any $y \in \mathcal{Y}$ and $E \in \mathscr{B}(\mathcal{Z})$. Thus, for any fixed set E taking the choice $y = \sqrt{B}\mu(E)$ implies that $(B\mu(E), \mu(E)) = 0$. Since E is arbitrary, this means $\mu = 0$ and thus finishes the proof for the sufficiency.

To prove the necessity, suppose first that G is not universal and hence, we know that, for some compact subset $\mathcal{Z}$ of $\mathcal{X}$, there exists a nonzero scalar measure $\nu \in \mathcal{M}(\mathcal{Z})$ such that $\nu \in \phi(\mathcal{Z})^\perp$, that is, $\int_{\mathcal{Z}} (\phi(x), w) d\nu(x) = 0$ for any $w \in \mathcal{W}$. This suggests to us to choose, for a nonzero $y_0 \in \mathcal{Y}$, the nonzero vector measure $\mu = y_0\nu$ defined by $\mu(E) := y_0\nu(E)$ for any $E \in \mathscr{B}(\mathcal{Z})$. Hence, the integral in Equation (21) equals

$$\int_{\mathcal{Z}} (\Phi^*(x)(w \otimes y), d\mu(x)) = (\sqrt{B}y, y_0) \int_{\mathcal{Z}} (\phi(x), w)_{\mathcal{W}} d\nu(x) = 0.$$

Therefore, we conclude that there exists a nonzero vector measure $\mu \in \Phi(\mathcal{Z})^\perp$, which implies that K is not universal.

If B is not positive definite, namely, there exists a nonzero element $y_1 \in \mathcal{Y}$ such that $(By_1, y_1) = 0$. However, we observe that $\|\sqrt{B}y_1\|^2 = (By_1, y_1)$ which implies that $\sqrt{B}y_1 = 0$. This suggests to us to choose a nonzero vector measure $\mu = y_1\nu$ with some nonzero scalar measure ν. Therefore, we conclude, for any $w \in \mathcal{W}$ and $y \in \mathcal{Y}$, that

$$\begin{aligned} \int_{\mathcal{Z}} (\Phi^*(x)(w \otimes y), d\mu(x)) &= (\sqrt{B}y, y_1) \int_{\mathcal{Z}} (\phi(x), w)_{\mathcal{W}} d\nu(x) \\ &= (y, \sqrt{B}y_1) \int_{\mathcal{Z}} (\phi(x), w)_{\mathcal{W}} d\nu(x) = 0, \end{aligned}$$

which implies that the nonzero vector measure $\mu \in \Phi(\mathcal{Z})^\perp$. This finishes the proof of the theorem. ■

In the special case $\mathcal{Y} = \mathbb{R}^n$, the operator B is an $n \times n$ positive semi-definite matrix. Then, Theorem 12 tells us that the matrix-valued kernel $K(x,t) := G(x,t)B$ is universal if and only if G is universal and the matrix B is of full rank.

We now proceed further and consider kernels produced by a finite combination of scalar kernels and operators. Specifically, we consider, for any $j \in \mathbb{N}_m$, that $G_j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a scalar kernel and $B_j \in \mathcal{L}_+(\mathcal{Y})$. We are interested in the kernel defined, for any $x,t \in \mathcal{X}$, by

$$K(x,t) := \sum_{j\in\mathbb{N}_m} G_j(x,t)B_j.$$

Suppose also, for each scalar kernel G_j, that there exists a Hilbert feature space $\mathcal{W}_j$ and a feature map $\phi_j : \mathcal{X} \to \mathcal{W}_j$.

To explain the associated feature map of kernel K, we need to define its feature space. For this purpose, let H_j be a Hilbert space with inner products $(\cdot,\cdot)_j$ for $j \in \mathbb{N}_m$ and we introduce the *direct sum Hilbert space* $\oplus_{j\in\mathbb{N}_m} H_j$ as follows. The elements in this space are of the form $(h_1,\ldots,h_m)$ with $h_j \in H_j$, and its inner product is defined, for any $(h_1,\ldots,h_m),(h'_1,\ldots,h'_m) \in \oplus_{j\in\mathbb{N}_m} H_j$, by

$$\langle (h_1,\ldots,h_m),(h'_1,\ldots,h'_m)\rangle := \sum_{j\in\mathbb{N}_m} (h_j,h'_j)_j.$$

This observation suggests to us to define the feature space of kernel K by the direct sum Hilbert space $\mathcal{W} := \oplus_{j\in\mathbb{N}_m}(\mathcal{W}_j \otimes \mathcal{Y})$, and its the map $\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{Y},\mathcal{W})$, for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, by

$$\Phi(x)y := (\phi_1(x) \otimes \sqrt{B_1}\, y, \ldots, \phi_m(x) \otimes \sqrt{B_m}\, y). \tag{22}$$

Hence, its adjoint operator $\Phi^* : \mathcal{X} \to \mathcal{L}(\oplus_{j\in\mathbb{N}_m}(\mathcal{W}_j \otimes \mathcal{Y}),\mathcal{Y})$ is given, for any $(w_1 \otimes y_1,\ldots,w_m \otimes y_m) \in \oplus_{j\in\mathbb{N}_m}(\mathcal{W}_j \otimes \mathcal{Y})$, by

$$\Phi^*(x)((w_1 \otimes y_1,\ldots,w_m \otimes y_m)) := \sum_{j\in\mathbb{N}_m} (\phi_j(x),w_j)_{\mathcal{W}_j} \sqrt{B_j}\, y_j.$$

Using the above observation, it is easy to see that, for any $x,t \in \mathcal{X}$, $K(x,t) = \Phi^*(x)\Phi(t)$. Thus K is a multi-task kernel and Φ is a feature map of K.

We are now in a position to state the result about the universality of the kernel K.

Theorem 13 *Suppose that $G_j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a continuous scalar universal kernel, and $B_j \in \mathcal{L}_+(\mathcal{Y})$ for $j \in \mathbb{N}_m$. Then, $K(x,t) := \sum_{j\in\mathbb{N}_m} G_j(x,t)B_j$ is universal if and only if $\sum_{j\in\mathbb{N}_m} B_j$ is positive definite.*

Proof Following Theorem 11, we need to prove that $\Phi(\mathcal{Z})^\perp = \{0\}$ for any compact set $\mathcal{Z}$ if and only if $\sum_{j\in\mathbb{N}_m} B_j$ is positive definite. To see this, observe that $\mu \in \Phi(\mathcal{Z})^\perp$ implies, for any $(w_1 \otimes y_1,\ldots,w_m \otimes y_m) \in \oplus_{j\in\mathbb{N}_m}(\mathcal{W}_j \otimes \mathcal{Y})$, that

$$\int_{\mathcal{Z}} \sum_{j\in\mathbb{N}_m} (\phi_j(x),w_j)_{\mathcal{W}_j}(\sqrt{B_j}\, y_j, d\mu(x)) = 0.$$

Since $w_j \in \mathcal{W}_j$ is arbitrary, the above equation is equivalent to

$$\int_{\mathcal{Z}} (\phi_j(x),w_j)_{\mathcal{W}_j}(\sqrt{B_j}\, y_j, d\mu(x)) = 0, \ \ \forall w_j \in \mathcal{W}_j, y_j \in \mathcal{Y} \text{ and } j \in \mathbb{N}_m, \tag{23}$$

which implies that $(\sqrt{B_j}\, y_j, d\mu(x)) \in \phi(\mathcal{Z})^\perp$ for any $j \in \mathbb{N}_m$. Recall that G_j is universal if and only if $\phi(\mathcal{Z})^\perp = \{0\}$. Therefore, Equation (23) holds true if and only if

$$(\mu(E), \sqrt{B_j}\, y_j) = (\sqrt{B_j}\mu(E), y_j) = 0, \quad \forall E \in \mathscr{B}(\mathcal{Z}), \ \ y_j \in \mathcal{Y}, j \in \mathbb{N}_m. \tag{24}$$

To move on to the next step, we will show that Equation (24) is true if and only if

$$(\mu(E), B_j\mu(E)) = 0, \quad \forall E \in \mathscr{B}(\mathcal{Z}), j \in \mathbb{N}_m. \tag{25}$$

To see this, we observe, for any $j \in \mathbb{N}_m$, that $\|\sqrt{B_j}\mu(E)\|^2 = (\mu(E), B_j\mu(E))$. Hence, Equation (25) implies Equation (24). Conversely, applying Equation (24) with the choice $y_j = \mu(E)$ directly yields Equation (25).

Moreover, we know, for any $y \in \mathcal{Y}$ and $j \in \mathbb{N}_m$, that $(B_j y, y)$ is nonnegative. Therefore, Equation (25) is equivalent to that

$$\big((\sum_{j\in\mathbb{N}_m} B_j)\mu(E), \mu(E)\big) = 0, \quad \forall E \in \mathscr{B}(\mathcal{Z}). \tag{26}$$

Therefore, we conclude that $\mu \in \Phi(\mathcal{Z})^\perp$ if and only if the above equation holds true.

Obviously, by Equation (26), we see that if $\sum_{j\in\mathbb{N}_m} B_j$ is positive definite then $\mu = 0$. This means that kernel K is universal. Suppose that $\sum_{j\in\mathbb{N}_m} B_j$ is not positive definite, that is, there exists a nonzero $y_0 \in \mathcal{Y}$ such that $\|\big(\sum_{j\in\mathbb{N}_m} B_j\big)^{\frac{1}{2}} y_0\|^2 := \big((\sum_{j\in\mathbb{N}_m} B_j)y_0, y_0\big) = 0$. Hence, choosing a nonzero vector measure $\mu := y_0\nu$, with ν a nonzero scalar measure, implies that Equation (26) holds true and, thus kernel K is not universal. This finishes the proof of the theorem. ■

Now we are in a position to analyze Examples 1 and 4 given in the Section 2. Since the function K considered in Example 1 is in the form of (22), we conclude that it is a multi-task kernel.

We now discuss a class of kernels which includes that presented in Example 4. To this end, we use the notation $\mathbb{Z}_+ = \{0\} \cup \mathbb{N}$ and, for any smooth function $f : \mathbb{R}^m \to \mathbb{R}$ and index $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}_+^m$, we denote the α-th partial derivative by $\partial^\alpha f(x) := \frac{\partial^{|\alpha|} f(x)}{\partial^{\alpha_1} x_1 \ldots \partial^{\alpha_m} x_m}$. Then, recall that the Sobolev space W^k with integer order k is the space of real valued functions with norm defined by

$$\|f\|_{W^k}^2 := \sum_{|\alpha|\le k} \int_{\mathbb{R}^m} |\partial^\alpha f(x)|^2 dx, \tag{27}$$

where $|\alpha| = \sum_{j\in\mathbb{N}_m} \alpha_j$, see Stein (1970). This space can be extended to any fractional index $s > 0$. To see this, we need the Fourier transform defined, for any $f \in L^1(\mathbb{R}^m)$, as

$$\hat{f}(\xi) := \int_{\mathbb{R}^m} e^{-2\pi i\langle x,\xi\rangle} f(x) dx, \qquad \forall \xi \in \mathbb{R}^m,$$

see Stein (1970). It has a natural extension to $L^2(\mathbb{R}^m)$ satisfying the Plancherel formula $\|f\|_{L^2(\mathbb{R}^m)} = \|\hat{f}\|_{L^2(\mathbb{R}^m)}$. In particular, we observe, for any $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{Z}_+^m$ and $\xi = (\xi_1, \ldots, \xi_m) \in \mathbb{R}^m$, that $\widehat{\partial^\alpha f}(\xi) = \hat{f}(\xi)(2\pi i\xi_1)^{\alpha_1} \ldots (2\pi i\xi_m)^{\alpha_m}$. Hence, by Plancherel formula, we see, for any $f \in W^k$ with $k \in \mathbb{N}$, that its norm $\|f\|_{W^k}$ is equivalent to

$$\Big(\int_{\mathbb{R}^m} (1 + 4\pi\|\xi\|^2)^k |\hat{f}(\xi)|^2 d\xi\Big)^{\frac{1}{2}}.$$

This observation suggests to us to introduce fractional Sobolev space W^s (see, e.g., Stein, 1970) with any order $s > 0$ with norm defined, for any function f, by

$$\|f\|_{W^s}^2 := \int_{\mathbb{R}^m} (1 + 4\pi^2\|\xi\|^2)^s |\hat{f}(\xi)|^2 d\xi.$$

Finally, we need the Sobolev embedding lemma which states that, for any $s > \frac{m}{2}$, there exists an absolute constant c such that, for any $f \in W^s$ and any $x \in \mathbb{R}^m$, there holds

$$|f(x)| \leq c\|f\|_{W^s},$$

(see, e.g., Folland, 1999; Stein, 1970).

The next result extends Example 4 to multivariate functions.

Proposition 14 *Let $\mathcal{Y} = L^2(\mathbb{R}^d)$, $\mathcal{X} = \mathbb{R}$ and $\mathcal{H}$ be the space of real-valued functions with norm*

$$\|f\|^2 := \int_{\mathbb{R}} \Big[\Big\| f(x,\cdot) \Big\|^2_{W^{\frac{d+1}{2}}} + \Big\| \frac{\partial f}{\partial x}(x,\cdot) \Big\|^2_{W^{\frac{d+1}{2}}} \Big] dx.$$

Then this is an RKHS with universal multi-task kernel given, for every $x,t \in \mathcal{X}$ by

$$(K(x,t)y)(r) = e^{-\pi|x-t|} \int_{\mathbb{R}^d} e^{-\pi\|r-s\|} y(s) ds, \;\; \forall y \in \mathcal{Y}, \; r \in \mathbb{R}^d. \tag{28}$$

Proof For any fixed $t \in \mathbb{R}^d$, it follows from the Sobolev embedding lemma that

$$|f(x,t)| \leq c\|f(\cdot,t)\|_{W^1}.$$

Combining this with the definition of Sobolev space W^1 given by Equation (27), we have that

$$\begin{aligned} \|f(x)\|^2_{\mathcal{Y}} &\leq c^2 \int_{\mathbb{R}^d} \|f(\cdot,t)\|^2_{W^1} dt \\ &= c^2 \int_{\mathbb{R}^d} \Big(\int_{\mathbb{R}} |f(x,t)|^2 + |\frac{\partial f}{\partial x}(x,t)|^2 dt \Big) dx \leq c^2 \|f\|^2. \end{aligned}$$

Since, for any $y \in \mathcal{B}_1$ and $x \in \mathbb{R}$, $|(y, f(x))_{\mathcal{Y}}| \leq \|y\|_{\mathcal{Y}} \|f(x)\|_{\mathcal{Y}} \leq \|f(x)\|_{\mathcal{Y}}$, by the above equation there exists a constant c' such that, for any $y \in \mathcal{B}_1, x \in \mathbb{R}$ and $f \in \mathcal{H}$,

$$|(y, f(x))_{\mathcal{Y}}| \leq c'\|f\|.$$

Hence, by the Riesz representation lemma, $\mathcal{H}$ is an RKHS (Micchelli and Pontil, 2005).

Next, we confirm Equation (28) is the kernel associated to $\mathcal{H}$. To this end, it suffices to show that the reproducing property holds, that is, for any $f \in \mathcal{H}$, $y \in \mathcal{Y}$ and $x \in \mathcal{X}$

$$(y, f(x))_{\mathcal{Y}} = \langle K_x y, f \rangle, \tag{29}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in $\mathcal{H}$.

By the Plancherel formula, we observe that the left-hand side of Equation (29) equals

$$\int_{\mathbb{R}^d} \hat{y}(\tau) \Big[\overline{\int_{\mathbb{R}} e^{2\pi i \langle x,\xi \rangle} \hat{f}(\xi,\tau) d\xi} \Big] d\tau = \int_{\mathbb{R}^d} \int_{\mathbb{R}} \hat{y}(\tau) e^{-2\pi i \langle x,\xi \rangle} \overline{\hat{f}(\xi,\tau)} d\xi d\tau.$$

On the other hand, note that $K_x y(x') := K(x',x)y \in \mathcal{Y}$, and consider its Fourier transform

$$(\widehat{K}(\cdot,x)y)(\xi,\tau) = \int_{\mathbb{R}^d} \int_{\mathbb{R}} e^{-2\pi i \langle x',\xi \rangle} e^{-2\pi i \langle r,\tau \rangle} (K(x',x)y)(r) dr dx'.$$

Using Equation (28) and the Plancherel formula, the integral on the right hand of the above equation is equal to

$$\frac{e^{-2\pi i\langle x,\xi\rangle}}{(1+4\pi^2|\xi|^2)}\frac{\hat{y}(\tau)}{(1+4\pi^2\|\tau\|^2)^{\frac{d+1}{2}}}. \tag{30}$$

However, the right-hand side of Equation (29) is identical to

$$\int_{\mathbb{R}^d}\int_{\mathbb{R}}(\widehat{K}(\cdot,x)y)(\xi,\tau)\overline{\hat{f}(\xi,\tau)}(1+4\pi^2|\xi|^2)(1+4\pi^2\|\tau\|^2)^{\frac{d+1}{2}}d\tau d\xi.$$

Putting (30) into the above equation, we immediately know that the reproducing property (29) holds true. This verifies that K is the reproducing kernel of the Hilbert space $\mathcal{H}$.

To prove the universality of this kernel, let $\mathcal{Z}$ be any prescribed compact subset of $\mathcal{X}$, we define the Laplace kernel, for any $x,t\in\mathbb{R}$, by $G(x,t):=e^{-|x-t|}$ and the operator $B:L^2(\mathbb{R}^d)\to L^2(\mathbb{R}^d)$ by

$$Bg(r):=\int_{\mathbb{R}^d}e^{-\|r-s\|}g(s)ds,\ \ \forall\, r\in\mathbb{R}^d.$$

Then, $K(x,t)=e^{-|x-t|}B$ and moreover

$$\widehat{Bg}(\tau)=c_d\frac{\hat{g}(\tau)}{(1+4\pi^2\|\tau\|^2)^{\frac{d+1}{2}}}. \tag{31}$$

By Theorem 12, it now suffices to prove that G is universal and B is positive definite. To this end, note that there exists c_d such that

$$G(x,t)=c_d\int_{\mathbb{R}}\frac{e^{-2\pi i\langle x-t,\xi\rangle}}{1+4\pi^2|\xi|^2}d\xi.$$

Since the weight function $\frac{1}{1+4\pi^2|\xi|^2}$ is positive, G is universal according to Micchelli et al. (2003).

To show the positive definiteness of B, we obtain from Equation (31) and the Plancherel formula that

$$(Bg,g)=c_d\int_{\mathbb{R}^d}\frac{|\hat{g}(\tau)|^2d\tau}{(1+4\pi^2\|\tau\|^2)^{\frac{d+1}{2}}}.$$

From this observation, the assertion follows. ■

We now discuss continuous parameterized multi-task kernels. For this purpose, let Ω be a locally compact Hausdorff space and, for any $\omega\in\Omega$, $B(\omega)$ be an $n\times n$ positive semi-definite matrix. We are interested in the kernel of the following form

$$K(x,t)=\int_{\Omega}G(\omega)(x,t)B(\omega)dp(\omega),\quad \forall x,t\in\mathcal{X}, \tag{32}$$

where p is a measure on Ω. We investigate this kernel in the case that, for any $\omega\in\Omega$, $G(\omega)$ is a scalar kernel with a feature representation given, for any $x,t\in\mathcal{X}$, by the formula $G(\omega)(x,t)=\langle\phi_\omega(x),\phi_\omega(t)\rangle_{\mathcal{W}}$. Now, we introduce the Hilbert space $\widetilde{\mathcal{W}}=L^2(\Omega,\mathcal{W}\otimes\mathcal{Y},p)$ with norm defined, for any $f:\Omega\to\mathcal{W}\otimes\mathcal{Y}$, by

$$\|f\|^2_{\widetilde{\mathcal{W}}}:=\int_{\Omega}\|f(\omega)\|^2_{\mathcal{W}\otimes\mathcal{Y}}dp(\omega).$$

Next, we define a map $\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{Y}, L^2(\Omega, \mathcal{W} \otimes \mathcal{Y}, p))$, for any $x \in \mathcal{X}$ and $\omega \in \Omega$, by

$$\Phi(x)y(\omega) := \phi_\omega(x) \otimes (\sqrt{B(\omega)}y).$$

By an argument similar to that used just before Theorem 13, we conclude that K is a multi-task kernel and has the feature map Φ with feature space $\widetilde{\mathcal{W}}$.

We are ready to present a sufficient condition on the universality of K.

Theorem 15 *Let p be a measure on Ω and for every ω in the support of p, let $G(\omega)$ be a continuous universal kernel and $B(\omega)$ a positive definite operator. Then, the multi-task kernel K defined by Equation (32) is universal.*

Proof Following Theorem 11, for a compact set $\mathcal{Z} \subseteq \mathcal{X}$ suppose that there exists a vector measure μ such that

$$\int_{\mathcal{Z}} \phi_\omega(x) \otimes \sqrt{B(\omega)}(d\mu(x)) = 0.$$

Therefore, there exists a $\omega_0 \in \text{support}(p)$ satisfying $\int_{\mathcal{Z}} \phi_{\omega_0}(x) \otimes \sqrt{B(\omega_0)}\big(d\mu(x)\big) = 0$. Equivalently, $\int_{\mathcal{Z}} \phi_{\omega_0}(x)\big(\sqrt{B(\omega_0)}d\mu(x), y\big) = 0$ for any $y \in \mathcal{Y}$. Since we assume $G(\omega_0)$ is universal, appealing to the feature characterization in the scalar case (Micchelli et al., 2006) implies that the scalar measure $(\sqrt{B(\omega_0)}d\mu(x), y) = 0$. Consequently, we obtain that $\mu \equiv 0$ since $y \in \mathcal{Y}$ is arbitrary. This completes the proof of this theorem. ■

Next we offer a concrete example of the above theorem.

Example 5 *Suppose the measure p over $[0,\infty)$ does not concentrate on zero and $B(\omega)$ be a positive definite $n \times n$ matrix for each $\omega \in (0,\infty)$. Then the kernel $K(x,t) = \int_0^\infty e^{-\omega\|x-t\|^2} B(\omega)dp(\omega)$ is a multi-task universal kernel.*

Further specializing this example, we choose the measure p to be the Lebesgue measure on $[0,\infty)$ and choose $B(\omega)$ in the following manner. Let A be $n \times n$ symmetric matrices. For every $\omega > 0$, we define the (i,j)-th entry of the matrix $B(\omega)$ as $e^{-\omega A_{ij}}$, $i, j \in \mathbb{N}_n$. Recall that a matrix A is *conditionally negative semi-definite* if, for any $c_i \in \mathbb{R}, i \in \mathbb{N}_n$ with $\sum_{i\in\mathbb{N}_n} c_i = 0$, then the quadratic form satisfies $\sum_{i,j\in\mathbb{N}_n} c_i A_{ij} c_j \leq 0$. A well-known theorem of I. J. Schoenberg (see, e.g., Micchelli, 1986) state that $B(\omega)$ is positive semi-definite for all $\omega > 0$ if and only if A is conditionally negative semi-definite. Moreover, if the elements of the conditionally negative semi-definite matrix A satisfy, for any $i, j \in \mathbb{N}_n$, the inequalities $A_{ij} > \frac{1}{2}(A_{ii} + A_{jj})$ and $A_{ii} > 0$, then $B(\omega)$ is positive definite (Micchelli, 1986). With this choice of A, the universal kernel in Example 5 becomes

$$\Big(K(x,t)\Big)_{ij} = \frac{1}{\|x-t\|^2 + A_{ij}}, \quad \forall i, j \in \mathbb{N}_n.$$

5.2 Transformation Kernels

In this subsection we explore matrix-valued kernels produced by transforming scalar kernels. To introduce this type of kernels, let $\mathcal{Y} = \mathbb{R}^n$, $\widetilde{\mathcal{X}}$ be a Hausdorff space and T_p be a map from $\mathcal{X}$ to

$\widetilde{\mathcal{X}}$ (not necessary linear) for $p \in \mathbb{N}_n$. Then, given a continuous scalar kernel $G : \widetilde{\mathcal{X}} \times \widetilde{\mathcal{X}} \to \mathbb{R}$, we consider the matrix-valued kernel on $\mathcal{X}$ defined by

$$K(x,t) := \Big(G(T_p x, T_q t)\Big)_{p,q=1}^{n}, \ \ \forall x,t \in \mathcal{X}. \tag{33}$$

Proposition 16 *Let G be a scalar kernel and K be defined by (33). Then, K is a matrix-valued kernel.*

Proof For any $m \in \mathbb{N}$, $\{y_i : y_i \in \mathbb{R}^n, i \in \mathbb{N}_m\}$ and $\{x_i : x_i \in \widetilde{\mathcal{X}}, i \in \mathbb{N}_m\}$ then

$$\sum_{i,j \in \mathbb{N}_m} (y_i, K(x_i,x_j) y_j) = \sum_{p,i} \sum_{q,j} y_{pi} y_{qj} G(T_p x_i, T_q x_j).$$

Since G is a scalar reproducing kernel on $\mathcal{Z}$, the last term of the above equality is nonnegative, and hence K is positive semi-definite matrix-valued kernel. This completes the proof of the assertion. ■

We turn our attention to the characterization of the universality of K defined by Equation (33). To this end, we assume that the scalar kernel G has a feature map $\phi : \widetilde{\mathcal{X}} \to \mathcal{W}$ and define the mapping $\Phi(x) : \mathbb{R}^n \to \mathcal{W}$, for any $y = (y_1, \ldots, y_n) \in \mathbb{R}^n$, by $\Phi(x)y = \sum_{p \in \mathbb{N}_n} y_p \phi(T_p x)$. Its adjoint operator $\Phi(x)^* : \mathcal{W} \to \mathbb{R}^n$ is given, for any $w \in \mathcal{W}$, as $\Phi^*(x)w = (\langle \phi(T_1 x), w \rangle_{\mathcal{W}}, \ldots, \langle \phi(T_n x), w \rangle_{\mathcal{W}})$. Then, for any $x,t \in \mathcal{X}$, the kernel $K(x,t) = \Phi^*(x)\Phi(t)$ and thus, we conclude that $\mathcal{W}$ is the feature space of K and Φ is its feature map.

We also need some further notation and definitions. For a map $T : \mathcal{X} \to \widetilde{\mathcal{X}}$, we denote its range space by $T\mathcal{X} := \{Tx : x \in \mathcal{X}\}$ and $T^{-1}(E) := \{x : Tx \in E\}$ for any $E \subset \widetilde{\mathcal{X}}$. In addition, we say that T is continuous if $T^{-1}(U)$ is open whenever U is a open set in $\widetilde{\mathcal{X}}$. Finally, for any scalar Borel measure ν on $\mathcal{X}$ and a continuous map T from $\mathcal{X}$ to $\widetilde{\mathcal{X}}$, we introduce the *image measure* $\nu \circ T^{-1}$ on $\widetilde{\mathcal{X}}$ defined, for any $E \in \mathscr{B}(\widetilde{\mathcal{X}})$, by $(\nu \circ T^{-1})(E) := \nu(\{x \in \mathcal{X} : Tx \in E\})$.

We are ready to state the result about universality of the kernel K in Equation (33).

Proposition 17 *Let G be a scalar universal kernel, $T_p : \mathcal{X} \to \widetilde{\mathcal{X}}$ be continuous for each $p \in \mathbb{N}_n$ and define the kernel K by Equation (33). Then K is universal if and only the sets $T_q\mathcal{X}$, $q \in \mathbb{N}_n$, are pairwise disjoint and T_q is one-to-one for each $q \in \mathbb{N}_n$.*

Proof Following Theorem 11, for any compact set $\mathcal{Z} \subseteq \mathcal{X}$, it suffices to verify the equation $\Phi(\mathcal{Z})^\perp = \{0\}$. Before doing so, we recall that, by Lemma 10 and the remark which followed it, for any vector measure $\mu \in \mathcal{M}(\mathcal{Z}, \mathbb{R}^n)$, there exists a scalar regular measure $\nu \in \mathcal{M}(\mathcal{Z} \times \mathcal{B}_1)$ such that

$$d\mu(t) = \Big(\int_{\mathcal{B}_1} y_1 d\nu(t,y), \ldots, \int_{\mathcal{B}_1} y_n d\nu(t,y) \Big).$$

Hence, any vector measure μ can be represented as $\mu = (\mu_1, \ldots, \mu_n)$ where each μ_i is a scalar measure. Then, $\mu \in \Phi(\mathcal{Z})^\perp$ can be rewritten as

$$\sum_{q \in \mathbb{N}_n} \int_{\mathcal{Z}} \phi(T_q t) d\mu_q(t) = 0.$$

Equivalently, if $\widetilde{\mathcal{Z}} := \cup_{q \in \mathbb{N}_n} T_q \mathcal{Z}$ we conclude that

$$\int_{\widetilde{\mathcal{Z}}} \phi(z) d\Big(\sum_{q \in \mathbb{N}_n} \mu_q \circ T_q^{-1} \Big)(z) = 0.$$

Since T_q is continuous for any $q \in \mathbb{N}_n$, the range space $T_q\mathcal{Z}$ is compact and so is $\widetilde{\mathcal{Z}}$. Recall from Micchelli et al. (2006) that the scalar kernel G is universal on $\widetilde{\mathcal{Z}}$ if and only if its feature map ϕ is universal on $\widetilde{\mathcal{Z}}$. Therefore, the above equation is reduced to the form

$$\sum_{q\in\mathbb{N}_n} \mu_q \circ T_q^{-1} = 0.$$

Consequently, we conclude that K is universal if and only if

$$\big\{(\mu_1,\ldots,\mu_n) : \sum_{q\in\mathbb{N}_n} \mu_q \circ T_q^{-1} = 0\big\} = \big\{0\big\}. \tag{34}$$

With the above derivation, we can now prove the necessity. Suppose that $\{T_q\mathcal{X} : q \in \mathbb{N}_n\}$ is not pairwise disjoint. Without loss of generality, we assume that $T_1\mathcal{X} \cap T_2\mathcal{X} \neq \emptyset$. That means there exists $x_1, x_2 \in \mathcal{X}$ such that $T_1x_1 = T_2x_2 = z_0$. Let $\mu_q \equiv 0$ for $q \geq 3$, and denote by $\delta_{x=x'}$ the point distribution at $x' \in \mathcal{X}$. Then, choosing $\mu_1 = \delta_{x=x_1}$, and $\mu_2 = -\delta_{x=x_2}$ implies that Equation (34) holds true. By Theorem 11 in Section 4, we know that K is not universal. This completes the first assertion.

Now suppose that there is a map, for example T_p, which is not one-to-one. This implies that there exists $x_1, x_2 \in \mathcal{X}$, $x_1 \neq x_2$, such that $T_px_1 = T_px_2$. Hence, if we let $\mu_q = 0$ for any $q \neq p$ and $\mu_p = \delta_{x=x_1} - \delta_{x=x_2}$ then $\sum_{q\in\mathbb{N}_n} \mu_q \circ T_q^{-1} = 0$,. But $\mu_p \neq 0$, hence, by Theorem 11, K is not universal. This completes the our assertion.

Finally, we prove the sufficiency. Since $\mu_q \circ T_q^{-1}$ only lives on $T_q\mathcal{X}$ and $\{T_q\mathcal{X} : q \in \mathbb{N}_n\}$ is pairwise disjoint, then $\sum_{q\in\mathbb{N}_n} \mu_q \circ T_q^{-1} = 0$ is equivalent to $\mu_q \circ T_q^{-1} = 0$ for each $q \in \mathbb{N}_n$. However, since T_q is one-to-one, $E = T_q^{-1}(T_q(E))$ for each Borel set $E \in \mathscr{B}(\mathcal{X})$. This means that $\mu_q(E) = \mu_q \circ T_q^{-1}(T_q(E)) = 0$ for any $E \in \mathscr{B}(\mathcal{X})$. This concludes the proof of the proposition. ■

We end this subsection with detailed proofs of our claims about the examples presented in Section 2. Indeed, we already proved the positive semi-definiteness of the kernel in Example 2 by Proposition 16. Below, we prove the claim that the function K given by Equation (6) is not a kernel in general.

Proposition 18 *Let $\sigma_{pq} > 0$ and $\sigma_{pq} = \sigma_{qp}$ for any $p,q \in \mathbb{N}_n$. Then, the matrix-valued function defined by*

$$K(x,t) := \left(e^{-\sigma_{pq}\|x-y\|^2}\right)_{p,q=1}^{n}, \ \ \forall x,t \in \mathcal{X}$$

is a multi-task kernel if and only if for some constant σ, $\sigma_{pq} = \sigma$ for any $p,q \in \mathbb{N}_n$.

Proof When $\left(\sigma_{pq}\right)_{p,q=1}^{n}$ is a constant matrix then K is positive semi-definite. Conversely, suppose K is a multi-task kernel which means, for any $m \in \mathbb{N}$ and $x_i \in \mathcal{X}$ with $i \in \mathbb{N}_m$, that the double-indexed $nm \times nm$ matrix

$$\left(G((i,p),(j,q)) = e^{-\sigma_{pq}\|x_i-x_j\|^2}\right)_{(i,p),(j,q)\in\mathbb{N}_m\times\mathbb{N}_n} \tag{35}$$

is positive semi-definite.

We choose any distinct positive integers p_0 and q_0. In Equation (35), we specify any m,n with $m \geq n$ such that $p_0, q_0 \in \mathbb{N}_m$, $x_1,\ldots,x_n$ with $x_{p_0} \neq x_{q_0}$ and set $c = \|x_{p_0} - x_{q_0}\|^2$. Therefore, we

conclude that the matrix

$$\begin{pmatrix} 1 & 1 & \exp\{-c\sigma_{p_0p_0}\} & \exp\{-c\sigma_{p_0q_0}\} \\ 1 & 1 & \exp\{-c\sigma_{q_0p_0}\} & \exp\{-c\sigma_{q_0q_0}\} \\ \exp\{-c\sigma_{p_0p_0}\} & \exp\{-c\sigma_{p_0q_0}\} & 1 & 1 \\ \exp\{-c\sigma_{q_0p_0}\} & \exp\{-c\sigma_{q_0q_0}\} & 1 & 1 \end{pmatrix}$$

is positive semi-definite. Consequently, the determinant of its 3×3 sub-matrix in the upper right hand corner, which equals $-\left[\exp\{-c\sigma_{p_0p_0}\}-\exp\{-c\sigma_{q_0p_0}\}\right]^2$, is nonnegative. Therefore, we conclude that $\sigma_{p_0p_0}=\sigma_{q_0p_0}$. ■

5.3 Hessian of Gaussian Kernels

In this subsection we consider the universal example of the Hessian of scalar Gaussian kernels (Example 3 in Section 2). To introduce this type of kernels, we let $\mathcal{X}=\mathcal{Y}=\mathbb{R}^n$ and G be a Gaussian kernel with deviation σ, that is, for any $x\in\mathbb{R}^n$, $G(x)=e^{-\|x\|^2/\sigma}$ with $\sigma>0$. Then, the Hessian matrix of the kernel G is

$$K(x,t):=\Big(-(\partial_p\partial_q G)(x-t)\Big)_{p,q=1}^{n}\quad\forall x,t\in\mathbb{R}^n. \tag{36}$$

and so alternatively, K has the form

$$K(x,t)=4\pi\big(2\pi\sigma\big)^{n/2}\int_{\mathbb{R}^n}e^{2\pi i\langle x-t,\xi\rangle}\xi\xi^T e^{-\sigma\|\xi\|^2}d\xi. \tag{37}$$

Corollary 19 *Let $n\geq 1$ and $K:\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}^{n\times n}$ be defined by (36). Then, K is a matrix-valued kernel which is universal if and only if $n=1$.*

Proof The fact that K is positive semi-definite directly follows from the observation, for any $m\in\mathbb{N}$, $\{y_i:y_i\in\mathbb{R}^n,i\in\mathbb{N}_m\}$ and $\{x_i:x_i\in\mathcal{X},i\in\mathbb{N}_m\}$, that

$$\sum_{i,j,\in\mathbb{N}_m}(y_i,K(x_i,x_j)y_j)=4\pi\big(2\pi\sigma\big)^{n/2}\int_{\mathbb{R}^n}\left|\sum_{i\in\mathbb{N}_m}\langle y_i,\xi\rangle e^{2\pi i\langle x_i,\xi\rangle}\right|^2 e^{-\sigma\|\xi\|^2}d\xi.$$

In order to prove the universality of K, we follow Theorem 11. For this purpose, we assume that $\mathcal{Z}$ is a compact subset of $\mathcal{X}$ and $\mu\in\mathcal{K}(\mathcal{Z})^\perp$, that is,

$$\int_{\mathcal{Z}}K(x,t)(d\mu(t))=0\quad\forall x\in\mathcal{Z}. \tag{38}$$

By Equation (37), this equation is equivalent to

$$\int_{\mathbb{R}^n}e^{2\pi i\langle x,\xi\rangle}\xi e^{-\sigma\|\xi\|^2}\int_{\mathcal{Z}}e^{-2\pi i\langle t,\xi\rangle}(\xi,d\mu(t))d\xi=0,\ \ \forall x\in\mathcal{Z},$$

which implies, by integrating both sides of this equation with respect to $x\in\mathbb{R}^n$, that

$$\int_{\mathbb{R}^n}e^{-\sigma\|\xi\|^2}\left|\int_{\mathcal{Z}}e^{-2\pi i\langle t,\xi\rangle}(\xi,d\mu(t))\right|^2 d\xi=0.$$

Consequently, Equation (38) is identical to the equation

$$\int_{\mathcal{Z}} e^{-2\pi i\langle t,\xi\rangle}(\xi, d\mu(t)) = 0, \qquad \forall \xi \in \mathbb{R}^n.$$

If $n = 1$, the above equation means that

$$\int_{\mathcal{Z}} e^{-2\pi i t\xi} d\mu(t) = 0, \qquad \forall \xi \in \mathbb{R}.$$

Taking the k-th derivative with respect to ξ of both sides of this equation and set $\xi = 0$, we have, for every $k \in \mathbb{N}$, that

$$\int_{\mathcal{Z}} t^k d\mu(t) = 0.$$

Since polynomials are dense in $\mathcal{C}(\mathcal{Z})$, we conclude from the above equation that $\mu = 0$. Hence, by Theorem 11, the kernel K is universal when $n = 1$.

If $n \geq 2$, we choose $\mu_q = 0$ for $q \geq 3$ and $d\mu_1(t) = dt_1(\delta_{t_2=1} - \delta_{t_2=-1})\prod_{p=3}^n \delta_{t_p=0}$ and $d\mu_2(t) = (\delta_{t_1=-1} - \delta_{t_1=1})dt_2 \prod_{p=3}^n \delta_{t_p=0}$, and note that

$$\int_{[-1,1]^n} e^{-2\pi i\langle t,\xi\rangle} d\mu_1(t) = (-2\pi i \sin(2\pi\xi_2))\frac{sin(2\pi\xi_1)}{\pi\xi_1},$$

and

$$\int_{[-1,1]^n} e^{-2\pi i\langle t,\xi\rangle} d\mu_2(t) = (2\pi i \sin(2\pi\xi_1))\frac{sin(2\pi\xi_2)}{\pi\xi_2}.$$

Therefore, we conclude that

$$\int_{[-1,1]^n} e^{-2\pi i\langle t,\xi\rangle}(\xi, d\mu(t)) = \xi_1 \int_{[-1,1]^n} e^{-2\pi i\langle t,\xi\rangle} d\mu_1(t) + \xi_2 \int_{[-1,1]^n} e^{-2\pi i\langle t,\xi\rangle} d\mu_2(t) = 0.$$

Hence, the kernel K is not universal when $n \geq 2$. ■

5.4 Projection Kernels

In the final subsection we introduce a class of multi-task kernels associated with projection operators of scalar kernels.

We start with some notation and definitions. Let $\mathcal{X} \subseteq \mathbb{R}^d$, $\Omega \subseteq \mathbb{R}^m$ be a compact set and $\mathcal{Y} = L^2(\Omega)$. We also need a continuous scalar kernel $G : (\mathcal{X} \times \Omega) \times (\mathcal{X} \times \Omega) :\to \mathbb{R}$ with a feature representation given, for any $x, x' \in \mathcal{X}$ and $t, s \in \Omega$, by

$$G((x,t),(x',s)) = \langle \phi(x,t), \phi(x',s)\rangle_{\mathcal{W}}. \tag{39}$$

Then, the projection kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{Y})$ is defined, for any $f \in L^2(\Omega)$, by

$$(K(x,x')f)(t) := \int_\Omega G((x,t),(x',s))f(s)ds, \;\; \forall\, x, x' \in \mathcal{X}, \;\; t \in \mathbb{R}. \tag{40}$$

We first show that K is a multi-task kernel. To see this, for any $m \in \mathbb{N}$, $\{x_i : x_i \in \mathcal{X}, i \in \mathbb{N}_m\}$, and $\{y_i : y_i \in L^2(\Omega), i \in \mathbb{N}_m\}$ there holds

$$\begin{aligned}\sum_{i,j\in\mathbb{N}_m}(K(x_i,x_j)y_j,y_i) &= \sum_{i,j\in\mathbb{N}_m}\int_\Omega\int_\Omega G((x_i,t),(x_j,s))y_j(s)y_i(t)dtds \\ &= \sum_{i\in\mathbb{N}_m}\left\|\int_\Omega \phi(x_i,s)y_i(s)ds\right\|^2 \geq 0,\end{aligned}$$

which implies that K is a kernel.

To investigate its universality from the feature perspective, we define the mapping $\Phi : \mathcal{X} \to \mathcal{L}(\mathcal{Y},\mathcal{W})$, for any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, by

$$\Phi(x)y := \int_\Omega \phi(x,s)y(s)ds,$$

and also its adjoint operator Φ^* is given, for any $w \in \mathcal{W}$, by $\Phi^*(x)w = \langle \phi(x,\cdot), w\rangle_{\mathcal{W}}$. Hence, for any $x,x' \in \mathcal{X}$, we conclude that $K(x,x') = \Phi^*(x)\Phi(x')$ which implies that K is a multi-task kernel and Φ is its associated feature map.

Our next goal is to prove the universality of K.

Theorem 20 *Let G and K be defined as in Equations* (39) *and* (40)*. If G is a universal scalar kernel then K is a universal multi-task kernel.*

Proof By Theorem 11, it suffices to show that, for any compact $\mathcal{Z} \subseteq \mathcal{X}$, whenever there exists a vector measure μ such that

$$\int_{\mathcal{Z}} \Phi(x)(d\mu(x)) = 0,$$

then $\mu = 0$. Note that μ is an $L^2(\Omega)$-valued measure. Hence, μ can alternatively be interpreted as a measure $\mu(\cdot,\cdot)$ on $\mathcal{Z}\times\Omega$ defined, for any $E \in \mathscr{B}(\mathcal{Z})$ and $E' \in \mathscr{B}(\Omega)$, by $\mu(E,E') := \int_{E'} \mu(E)(s)ds$. From this observation, we know that

$$\int_{\mathcal{Z}} \Phi(x)(d\mu(x)) = \int_{\mathcal{Z}}\int_\Omega \phi(x,s)d\mu(x,s).$$

Since $\mathcal{Z}$ and Ω are both compact, then $\mathcal{Z}\times\Omega$ is also compact by Tychonoff theorem (Folland, 1999, p.136). By assumption, G is universal on $\mathcal{X}\times\Omega$ and ϕ is its feature map, and thus we conclude that the scalar measure $d\mu(x,s)$ is the zero measure. This means that, for any $E \in \mathscr{B}(\mathcal{Z})$ and $E' \in \mathscr{B}(\Omega)$,

$$\int_{E'} \mu(E)(s)ds = 0.$$

Since E, E' are arbitrary, we conclude that the vector measure $\mu = 0$ which completes the assertion. ■

6. Conclusion

We have presented a characterization of multi-task kernels from a Functional Analysis perspective. Our main result, Theorem 4 established the equivalence between two spaces associated with the kernel. The first space, $\mathcal{C}_K(\mathcal{Z},\mathcal{Y})$, is the closure of the linear span of kernel sections; the second space, $\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$, is the closure of the linear span of the features associated with the kernel. In both cases, the closure was relative to the Banach space of continuous vector-valued functions. This result is important in that it allows one to verify the universality of a kernel directly by considering its features.

We have presented two alternate proofs of Theorem 4. The first proof builds upon the work of Micchelli et al. (2006) and the observation that a multi-task kernel can be reduced to a standard scalar-valued kernel on the cross product space $\mathcal{Z}\times\mathcal{Y}$. The second proof relies upon the theory of vector-measures. This proof is constructive and provides necessary and sufficient conditions on the universality of a multi-task kernel. They are summarized in Theorem 11, which is our main tool for verifying the universality of a multi-task kernel.

In both proofs, an important ingredient is a principle from Functional Analysis, which uses the notion of the *annihilator* set. This principle, which is a consequence of the Hahn-Banach Theorem, states that two closed linear subspaces of a Banach space—in our case $\mathcal{C}_K(\mathcal{Z},\mathcal{Y})$ and $\mathcal{C}_\Phi(\mathcal{Z},\mathcal{Y})$—are equal if and only if whenever a bounded linear functional vanishes on one of them, it also vanishes on the other one.

A substantial part of the paper has been devoted to present several examples of multi-task kernels, some of which are valuable for applications. Although much remains to be done on developing applications of the theory of universal kernels, we hope that our theoretical findings, as they are illustrated through the examples, will motivate further work on multi-task learning in applied machine learning.

Acknowledgments

This work was supported by EPSRC Grants GR/T18707/01 and EP/D052807/1 by the IST Programme of the European Community, under the PASCAL Network of Excellence IST-2002-506778. The first author was supported by the NSF grant 0325113, the FIRB project RBIN04PARL, the EU Integrated Project Health-e-Child IST-2004-027749, and the City University of Hong Kong grant No.7200111(MA). The second author is supported by NSF grant DMS 0712827.

We are grateful to Alessandro Verri, Head of the Department of Computer Science at the University of Genova for providing us with the opportunity to complete part of this work in a scientifically stimulating and friendly environment. We also wish to thank the referees for their valuable comments.

Appendix A.

This appendix gives the proof of Lemmas 9 and 10 in Section 4.

Proof of Lemma 9 By the definition of the integral appearing in the right-hand side of equation it follows (17) (see, e.g., Diestel and Uhl, Jr., 1977), for any $f \in \mathcal{C}(\mathcal{Z},\mathcal{Y})$, that

$$|L_\mu f| \leq \|\mu\| \sup_{x\in\mathcal{Z}} \|f(x)\|_{\mathcal{Y}}.$$

Therefore, we obtain that $\|L_\mu\| \leq \|\mu\|$, and thus $L_\mu \in \mathcal{C}^*(\mathcal{Z},\mathcal{Y})$.

To show that $\|L\| = \|\mu\|$, it remains to establish that $\|\mu\| \leq \|L_\mu\|$. To this end, for any $\varepsilon > 0$ and, by the definition of $\|\mu\|$, we conclude that there exist pairwise disjoint sets $\{A_j : j \in \mathbb{N}_n\}$ such that $\cup_{j\in\mathbb{N}_n} A_j \subseteq \mathcal{Z}$ and $\|\mu\| := |\mu|(\mathcal{Z}) \leq \varepsilon + \sum_{j\in\mathbb{N}_n} \|\mu(A_j)\|_{\mathcal{Y}}$. We introduce the function $g = \sum_{j\in\mathbb{N}_n} \frac{\mu(A_j)}{\|\mu(A_j)\|_{\mathcal{Y}}}\chi_{A_j}$ which satisfies, for any $x \in \mathcal{Z}$, the bound $\|g(x)\|_{\mathcal{Y}} \leq 1$. Since $|\mu|$ is a regular measure on $\mathcal{Z}$, applying Lusin's theorem (Folland, 1999, p.217) to the function χ_{A_j}, there exists a real-valued continuous function $f_j \in \mathcal{C}(\mathcal{Z})$ such that $|f_j(x)| \leq 1$ for any $x \in \mathcal{Z}$ and $f_j = \chi_{A_j}$, except on a set E_j with $|\mu|(E_j) \leq \frac{\varepsilon}{(n+1)2^{j+1}}$. We now define a function $h : \mathcal{Y} \to \mathcal{Y}$ by setting $h(y) = y$, if $\|y\|_{\mathcal{Y}} \leq 1$ and $h(y) = \frac{y}{\|y\|_{\mathcal{Y}}}$, if $\|y\|_{\mathcal{Y}} \geq 1$, and introduce another function in $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ given by $\bar{f} := \sum_{j\in\mathbb{N}_n} \frac{\mu(A_j)}{\|\mu(A_j)\|_{\mathcal{Y}}} f_j$. Therefore, the function $f = h \circ \bar{f}$ is in $C(\mathcal{Z},\mathcal{Y})$ as well, because $\bar{f} \in \mathcal{C}(\mathcal{Z},\mathcal{Y})$ and, for any $y,y' \in \mathcal{Y}$, $\|h(y) - h(y')\|_{\mathcal{Y}} \leq 2\|y - y'\|_{\mathcal{Y}}$. Moreover, we observe, for any $x \in \left(\cup_{j\in\mathbb{N}_n} E_j\right)^c$, that $f(x) = g(x)$ and, for any $x \in \mathcal{Z}$, that $\|f(x)\|_{\mathcal{Y}} \leq 1$.

We are now ready to estimate the total variation of μ. First, observe that

$$\int_{\mathcal{Z}} \|f(x) - g(x)\|_{\mathcal{Y}}\, d|\mu|(x) \leq \sum_{j\in\mathbb{N}_n} (n+1)|\mu|(E_j) \leq \varepsilon,$$

and consequently we obtain the inequality

$$\begin{aligned}
\|\mu\| &\leq \textstyle\sum_{j\in\mathbb{N}_n} \|\mu(A_j)\|_{\mathcal{Y}} + \varepsilon = \int_{\mathcal{Z}} (g(x), d\mu(x)) + \varepsilon \\
&\leq \left| \int_{\mathcal{Z}} (f(x) - g(x), d\mu(x)) \right| + \left| \int_{\mathcal{Z}} (f(x), d\mu(x)) \right| + \varepsilon \\
&\leq \int_{\mathcal{Z}} \|f(x) - g(x)\|_{\mathcal{Y}} d|\mu|(x) + \|L_\mu\| + \varepsilon \leq 2\varepsilon + \|L_\mu\|.
\end{aligned}$$

This finishes the proof of the lemma. ■

We proceed to the proof of Lemma 10.

Proof of Lemma 10 For each $\mu \in \mathcal{M}(\mathcal{X},\mathcal{Y})$, there exists an $L_\mu \in \mathcal{C}^*(\mathcal{Z},\mathcal{Y})$ given by Equation (17). The isometry of the map $\mu \mapsto L_\mu$ follows from Lemma 9.

Therefore, it suffices to prove, for every $\bar{L} \in \mathcal{C}^*(\mathcal{Z},\mathcal{Y})$, that there is a $\mu \in \mathcal{M}(\mathcal{Z},\mathcal{Y})$ such that $L_\mu = \bar{L}$. To this end, note that $\bar{L} \circ \iota^{-1} \in \mathcal{C}_\iota^*(\mathcal{Z}\times\mathcal{B}_1)$ since ι is an isometric map from $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ onto $\mathcal{C}_\iota(\mathcal{Z}\times\mathcal{B}_1)$ defined by Equation (10). Since $\mathcal{C}_\iota(\mathcal{Z}\times\mathcal{B}_1)$ is a closed subspace of $\mathcal{C}(\mathcal{Z}\times\mathcal{B}_1)$, applying the *Hahn-Banach extension theorem* (see, e.g., Folland, 1999, p.222) yields that, for any $L \in \mathcal{C}_\iota^*(\mathcal{Z}\times\mathcal{B}_1)$, there exists an extension functional $\tilde{L} \in \mathcal{C}^*(\mathcal{Z}\times\mathcal{B}_1)$ such that $\tilde{L}(F) = L \circ \iota^{-1}(F)$ for any $F \in \mathcal{C}_\iota(\mathcal{Z}\times\mathcal{B}_1)$. Moreover, recalling that $\mathcal{Z}\times\mathcal{B}_1$ is compact if $\mathcal{B}_1$ is equipped with the weak topology, by the Riesz representation theorem, for any $\tilde{L}$, there exists a scalar measure ν on $\mathcal{Z}\times\mathcal{B}_1$ such that

$$\tilde{L}(F) = \int_{\mathcal{Z}\times\mathcal{B}_1} F(x,y) d\nu(x,y), \quad \forall F \in \mathcal{C}(\mathcal{Z}\times\mathcal{B}_1).$$

Equivalently, for any $f \in \mathcal{C}(\mathcal{Z},\mathcal{Y})$ there holds

$$\bar{L}f = \bar{L}\circ\iota^{-1}(F) = \int_{\mathcal{Z}\times\mathcal{B}_1} F(x,y)d\nu(x,y) = \int_{\mathcal{Z}} (f(x),d\mu(x)) = L_\mu f,$$

where μ is defined i terms of ν as in Equation (15).

This finishes the identification between functionals in $\mathcal{C}(\mathcal{Z},\mathcal{Y})$ and vector measures with bounded variation. ■

References

L. Amodei. Reproducing kernels of vector–valued function spaces. In *Proc. of Chamonix*, A. Le Meehaute et al. Eds., pages 1–9, 1997.

N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.* 68:337–404, 1950.

S. K. Berberian. *Notes on Spectral Theory*. New York: Van Nostrand, 1966.

J. Burbea and P. Masani. *Banach and Hilbert Spaces of Vector-Valued Functions*. Pitman Research Notes in Mathematics Series, 90, 1984.

A. Caponnetto and E. De Vito. Optimal rates for regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7:331–368, 2007.

C. Carmeli, E. De Vito, and A. Toigo. Vector valued reproducing kernel Hilbert spaces of integrable functions and Mercer theorem. *Analysis and Applications*, 4:377–408, 2006.

F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39:1–49, 2001.

D. R. Chen, Q. Wu, Y. Ying, and D.X. Zhou. Support vector machine soft margin classifiers: error analysis. *Journal of Machine Learning Research*, 5:1143–1175, 2004.

A. Devinatz. On measurable positive definite operator functions. *J. Lonon Math. Soc.*, 35:417–424, 1960.

N. Dinculeanu. *Vector Measures*. Pergamon, Berlin, 1967.

J. Diestel and J. J. Uhl, Jr. *Vector Measures*. AMS, Providence (Math Surveys 15), 1977.

T. Evgeniou, C. A. Micchelli and M. Pontil. Learning multiple tasks with kernel methods. *J. Machine Learning Research*, 6:615–637, 2005.

G. B. Folland. *Real Analysis: Modern Techniques and Their Applications*. 2nd edition, New York, John Wiley & Sons, 1999.

A. Gretton, K.M. Borgwardt, M. Rasch, B. Schölkopf and A.J. Smola. A kernel method for the two-sample problem. In *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt and T. Hoffman editors, pages 513–520, MIT Press, 2007.

P. Lax. *Functional Analysis*., John Wiley & Sons, 2002.

S. Lowitzsch. A density theorem for matrix-valued radial basis functions. *Numerical Algorithms*, 39:253-256, 2005.

C. A. Micchelli, Interpolation of scattered data: distances matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

C. A. Micchelli and M. Pontil. A function representation for learning in Banach spaces. In *Proceedings of the 17th Annual Conference on Learning Theory (COLT'04)*, pages 255–269, 2004.

C. A. Micchelli and M. Pontil. On leaning vector-valued functions. *Neural Computation*, 17:177-204, 2005.

C.A. Micchelli and M. Pontil. Feature space perspectives for learning the kernel. *Machine Learning*, 66:297–319, 2007.

C. A. Micchelli, Y. Xu, and P. Ye. Cucker Smale learning theory in Besov spaces. *NATO Science Series sub Series III Computer and System Science*, 190:47–68, 2003.

C. A. Micchelli, Y. Xu, and H. Zhang. Universal kernels. *J. Machine Learning Research*, 7:2651-2667, 2006.

S. Mukherjee and D.X. Zhou. Learning coordinate covariances via gradients, *J. of Machine Learning Research* 7:519-549, 2006.

T. Poggio, S. Mukherjee, R. Rifkin, A. Rakhlin, and A. Verri. b. In *Uncertainty in Geometric Computations*, J. Winkler and M. Niranjan (eds.), Kluwer, 131–141, 2002.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*, MIT Press, 2006.

M. Reisert and H. Burkhardt. Learning equivariant functions with matrix valued kernels. *J. Machine Learning Research*, 8:385–408, 2007.

B. Schölkopf and A. J. Smola. *Learning with Kernels.* The MIT Press, Cambridge, MA, USA, 2002.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

E. Solak, R. Murray-Smith, W.E. Leithead, D.J. Leith and C.E. Rasmussen. Derivative observations in Gaussian Process models of dynamic Systems. In *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun and K. Obermayer editors, pages 1033–1040, MIT Press, 2003.

E. M. Stein. *Singular Integrals and Differential Properties of Functions*, Princeton University Press, Princeton, NJ, 1970.

I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *J. Machine Learning Research*, 2:67–93, 2001.

I. Steinwart, D. Hush, and C. Scovel. Function classes that approximate the Bayes risk. In *Proceeding of the 19th Annual Conference on Learning Theory*, pages 79–93, 2006.

K. Yosida. *Functional Analysis*, 6th edition, Springer-Verlag, 1980.

E. Vazquez and E. Walter. Multi-output support vector regression. In *Proceedings of the 13th IFAC Symposium on System Identification*, 2003.

D. X. Zhou. Density problem and approximation error in learning theory. Preprint, 2003.

Journal of Machine Learning Research 9 (2008) 1647-1678 Submitted 4/07; Revised 2/08; Published 8/08

A New Algorithm for Estimating the Effective Dimension-Reduction Subspace

Arnak S. Dalalyan ARNAK.DALALYAN@UPMC.FR
Université Paris 6 - Pierre et Marie Curie
Laboratoire de Probabilités, B. C. 188
75252 Paris Cedex 05, France

Anatoly Juditsky ANATOLI.IOUDITSKI@IMAG.FR
University Joseph Fourier of Grenoble LMC-IMAG
51 rue des Mathematiques, B. P. 53
38041 Grenoble Cedex 9, France

Vladimir Spokoiny SPOKOINY@WIAS-BERLIN.DE
Weierstrass Institute for Applied Analysis and Stochastics
Mohrenstrasse 39, 10117 Berlin Germany

Editor: Aapo Hyvarinen

Abstract

The statistical problem of estimating the effective dimension-reduction (EDR) subspace in the multi-index regression model with deterministic design and additive noise is considered. A new procedure for recovering the directions of the EDR subspace is proposed. Many methods for estimating the EDR subspace perform principal component analysis on a family of vectors, say $\hat{\beta}_1,\dots,\hat{\beta}_L$, nearly lying in the EDR subspace. This is in particular the case for the structure-adaptive approach proposed by Hristache et al. (2001a). In the present work, we propose to estimate the projector onto the EDR subspace by the solution to the optimization problem

$$\text{minimize} \quad \max_{\ell=1,\dots,L} \hat{\beta}_\ell^\top (I-A)\hat{\beta}_\ell \quad \text{subject to} \ \ A \in \mathcal{A}_{m^*},$$

where $\mathcal{A}_{m^*}$ is the set of all symmetric matrices with eigenvalues in $[0,1]$ and trace less than or equal to m^*, with m^* being the true structural dimension. Under mild assumptions, $\sqrt{n}$-consistency of the proposed procedure is proved (up to a logarithmic factor) in the case when the structural dimension is not larger than 4. Moreover, the stochastic error of the estimator of the projector onto the EDR subspace is shown to depend on L logarithmically. This enables us to use a large number of vectors $\hat{\beta}_\ell$ for estimating the EDR subspace. The empirical behavior of the algorithm is studied through numerical simulations.

Keywords: dimension-reduction, multi-index regression model, structure-adaptive approach, central subspace

1. Introduction

One of the most challenging problems in modern statistics is to find efficient methods for treating high-dimensional data sets. In various practical situations the problem of predicting or explaining a scalar response variable Y by d scalar predictors $X^{(1)},\dots,X^{(d)}$ arises. For solving this problem one should first specify an appropriate mathematical model and then find an algorithm for estimating that model based on the observed data. In the absence of a priori information on the relationship

between Y and $X = (X^{(1)}, \dots, X^{(d)})$, complex models are to be preferred. Unfortunately, the accuracy of estimation is in general a decreasing function of the model complexity. For example, in the regression model with additive noise and two-times continuously differentiable regression function $f : \mathbb{R}^d \to \mathbb{R}$, the most accurate estimators of f based on a sample of size n have a quadratic risk decreasing as $n^{-4/(4+d)}$ when n becomes large. This rate deteriorates very rapidly with increasing d leading to unsatisfactory accuracy of estimation for moderate sample sizes. This phenomenon is called "curse of dimensionality", the latter term being coined by Bellman (1961).

To overcome the "curse of dimensionality", additional restrictions on the candidates f for describing the relationship between Y and X are necessary. One popular approach is to consider the multi-index model with m^* indices: for some linearly independent vectors ϑ_1, ..., ϑ_{m^*} and for some function $g : \mathbb{R}^{m^*} \to \mathbb{R}$, the relation $f(x) = g(\vartheta_1^\top x, \dots, \vartheta_{m^*}^\top x)$ holds for every $x \in \mathbb{R}^d$. Here and in the sequel the vectors are understood as one column matrices and $M^\top$ denotes the transpose of the matrix M. Of course, such a restriction is useful only if $m^* < d$ and the main argument in favor of using the multi-index model is that for most data sets the underlying structural dimension m^* is substantially smaller than d. Therefore, if the vectors ϑ_1, ..., ϑ_{m^*} are known, the estimation of f reduces to the estimation of g, which can be performed much better because of lower dimensionality of the function g compared to that of f.

Another advantage of the multi-index model is that it postulates that only few linear combinations of the predictors may suffice for "explaining" the response Y. Considering these combinations as new predictors leads to a much simpler model (due to its low dimensionality), which can be successfully analyzed by graphical methods, see Cook and Weisberg (1999) and Cook (1998) for more details.

Throughout this work we assume that we are given n observations $(Y_1, X_1), \dots, (Y_n, X_n)$ from the model

$$Y_i = f(X_i) + \varepsilon_i = g(\vartheta_1^\top X_i, \dots, \vartheta_{m^*}^\top X_i) + \varepsilon_i, \tag{1}$$

where $\varepsilon_1, \dots, \varepsilon_n$ are unobserved errors assumed to be mutually independent zero mean random variables, independent of the design $\{X_i, i \le n\}$.

Since it is unrealistic to assume that $\vartheta_1, \dots, \vartheta_{m^*}$ are known, estimation of these vectors from the data is of high practical interest. When the function g is unspecified, only the linear subspace $\mathcal{S}_\vartheta$ spanned by these vectors may be identified from the sample. This subspace is usually called *index space* or *dimension-reduction (DR) subspace*. Clearly, there are many DR subspaces for a fixed model f. Even if f is observed without error, only the smallest DR subspace, henceforth denoted by $\mathcal{S}$, can be consistently identified. This smallest DR subspace, which is the intersection of all DR subspaces, is called *effective dimension-reduction* (EDR) subspace (Li, 1991) or *central mean subspace* (Cook and Li, 2002). We adopt in this paper the former term, in order to be consistent with Hristache et al. (2001a) and Xia et al. (2002), which are the closest references to our work.

The present work is devoted to studying a new algorithm for estimating the EDR subspace. We call it structural adaptation via maximum minimization (SAMM). It can be regarded as a branch of the structure-adaptive (SA) approach introduced in Hristache et al. (2001b) and Hristache et al. (2001a).

Note that a closely related problem is the estimation of the central subspace (CS), see Cook and Weisberg (1999) for its definition. For model (1) with i.i.d. predictors, the CS coincides with the EDR subspace. Hence, all the methods developed for estimating the CS can potentially be applied in our set-up. We refer to Cook and Li (2002) for background on the difference between the CS and

the central mean subspace and to Cook and Ni (2005) for a discussion of the relationship between different algorithms estimating these subspaces.

There are a number of methods providing an estimator of the EDR subspace in our set-up. These include ordinary least square (Li and Duan, 1989), sliced inverse regression (Li, 1991), sliced inverse variance estimation (Cook and Weisberg, 1991), principal Hessian directions (Li, 1992), graphical regression (Cook, 1998), parametric inverse regression (Bura and Cook, 2001a), SA approach (Hristache et al., 2001a), iterative Hessian transformation (Cook and Li, 2002), minimum average variance estimation (MAVE) (Xia et al., 2002), nonparametric linear smoothing for inverse regression (Bura, 2003), minimum discrepancy approach (Cook and Ni, 2005) marginal high moment regression (Yin and Cook, 2006) density based MAVE and outer product of gradient (Xia, 2007), as well as the refinements using contour-projection (Wang et al., 2008), intraslice covariance estimation (Cook and Ni, 2006) and Lasso shrinkage (Ni et al., 2005; Li, 2007).

All these methods, except SA approach and MAVE, rely on the principle of inverse regression (IR). Therefore they inherit its well known limitations. First, they require a hypothesis on the probabilistic structure of the predictors usually called linearity condition. Second, there is no theoretical justification guaranteeing that these methods estimate the whole EDR subspace and not just a part thereof, see Cook and Li (2004, Section 3.1) and the comments on the third example in Hristache et al. (2001a, Section 4). In the same time, they have the advantage of being simple for implementation and for inference.

The two other methods mentioned above—SA approach and MAVE—have much wider applicability including even time series analysis. The inference for these methods is more involved than that of IR based methods, but SA approach and MAVE need no strong requirements on the design of covariates or on the response variable. Moreover, in many cases they provide more accurate estimates of the EDR subspace (Hristache et al., 2001a; Xia et al., 2002; Xia, 2007).

These arguments, combined with empirical experience, indicate the complementarity of different methods designed to estimate the EDR subspace. It turns out that there is no procedure among those cited above that outperforms all the others in plausible settings. Therefore, a reasonable strategy for estimating the EDR subspace is to execute different procedures and to take a decision after comparing the obtained results. In the case of strong contradictions, collecting additional data is recommended.

The algorithm SAMM we introduce here exploits the fact that the gradient ∇f of the regression function f evaluated at any point $x \in \mathbb{R}^d$ belongs to the EDR subspace. The estimation of the gradient being an ill-posed inverse problem, it is better to estimate some linear combinations of $\nabla f(X_1), \ldots, \nabla f(X_n)$, which still belong to the EDR subspace.

Let L be a positive integer. The main idea behind the algorithm proposed in Hristache et al. (2001a) consists in iteratively estimating L linear combinations $\beta_1, \ldots, \beta_L$ of vectors $\nabla f(X_1), \ldots, \nabla f(X_n)$ and then recovering the EDR subspace from the vectors β_ℓ by running a principal component analysis (PCA). The resulting estimator is proved to be $\sqrt{n}$-consistent provided that L is chosen independently of the sample size n. Unfortunately, if L is small with respect to n, the subspace spanned by the vectors $\beta_1, \ldots, \beta_L$ may cover only a part of the EDR subspace. Therefore, empirical experience advocates for large values of L, even if the desirable feature of $\sqrt{n}$-consistency fails in this case.

The estimator proposed in the present work is designed to provide a remedy for this dissension between the theory and empirical experience. This goal is achieved by introducing a new method of extracting the EDR subspace from the estimators of the vectors $\beta_1, \ldots, \beta_L$. If we think of PCA

as the solution to a minimization problem involving a sum over L terms, see (5) in the next section, then, to some extent, our proposal consists in replacing the sum by the maximum. This motivates the term structural adaptation via maximum minimization. The main advantage of SAMM is that it allows us to deal with the case when L increases polynomially in n and yields an estimator of the EDR subspace which is consistent under a very weak identifiability assumption. In addition, SAMM provides a $\sqrt{n}$-consistent estimator (up to a logarithmic factor) of the EDR subspace when $m^* \leq 4$.

If $m^* = 1$, the corresponding model is referred to as *single-index* regression. There are many methods for estimating the EDR subspace in this case, see Yin and Cook (2005); Delecroix et al. (2006) and the references therein. Note also that the methods for estimating the EDR subspace have often their counterparts in the partially linear regression analysis, see for example Samarov et al. (2005) and Chan et al. (2004).

An interesting problem in the context of dimensionality reduction is the estimation of the true structural dimension m^*. Many approaches exist for constructing estimators of m^*, see (Li, 1991, Section 5), (Xia et al., 2002, Section 2.2), and Bura and Cook (2001b), Bura and Cook (2001a), Bura (2003) and Cook and Li (2004) and the references therein. Here we assume that the structural dimension is known, leaving the development of an extension to the case of unknown m^* for future investigation.

The rest of the paper is organized as follows. We review the structure-adaptive approach and introduce the SAMM procedure in Section 2. Theoretical features including $\sqrt{n}$-consistency of the procedure are stated in Section 3. Section 4 contains an empirical study of the proposed procedure through Monte Carlo simulations. The technical proofs are deferred to Section 5.

2. Structural Adaptation and SAMM

Introduced in Hristache et al. (2001b), the structure-adaptive approach is based on two observations. First, knowing the structural information helps better estimate the model function. Second, improved model estimation contributes to recovering more accurate structural information about the model. These advocate for the following iterative procedure. Start with the null structural information, then iterate the above-mentioned two steps (estimation of the model and extraction of the structure) several times improving the quality of model estimation and increasing the accuracy of structural information during the iteration.

2.1 Purely Nonparametric Local Linear Estimation

When no structural information is available, one can only proceed in a fully nonparametric way. A proper estimation method is based on local linear smoothing (cf. Fan and Gijbels, 1996, for more details): estimators of the function f and its gradient ∇f at a point X_i are given by

$$\begin{aligned}\begin{pmatrix} \hat{f}(X_i) \\ \widehat{\nabla f}(X_i) \end{pmatrix} &= \operatorname*{arg\,min}_{(a,c)^\top} \sum_{j=1}^{n} \big(Y_j - a - c^\top X_{ij}\big)^2 K\big(|X_{ij}|^2/b^2\big) \\ &= \left\{ \sum_{j=1}^{n} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix}^\top K\left(\frac{|X_{ij}|^2}{b^2}\right) \right\}^{-1} \sum_{j=1}^{n} Y_j \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} K\left(\frac{|X_{ij}|^2}{b^2}\right),\end{aligned}$$

where $X_{ij} = X_j - X_i$, b is a *bandwidth* and $K(\cdot)$ is a univariate kernel supported on $[0,1]$. (For a vector v, $|v|$ stands for its Euclidean norm.) The bandwidth b should be selected so that the ball

with radius b centered at the point of estimation X_i contains at least $d+1$ design points. For large value of d this leads to a large bandwidth and to a strong estimation bias. The goal of the structural adaptation is to diminish this bias using an iterative procedure exploiting the available estimated structural information.

In order to transform these general observations into a concrete procedure, let us describe in the rest of this section how the knowledge of the structure can help to improve the quality of the estimation and how the structural information can be obtained when the function or its estimator is given.

2.2 Model Estimation When an Estimator of $\mathcal{S}$ is Available

Let us start with the case of known $\mathcal{S}$. The function f has the same smoothness as g in the directions of the EDR subspace $\mathcal{S}$ spanned by the vectors $\vartheta_1,\dots,\vartheta_{m^*}$, whereas it is constant (and therefore, infinitely smooth) in all the orthogonal directions. This suggests to apply an anisotropic bandwidth for estimating the model function and its gradient. The corresponding local-linear estimator can be defined by

$$\begin{pmatrix} \hat{f}(X_i) \\ \widehat{\nabla f}(X_i) \end{pmatrix} = \left\{ \sum_{j=1}^{n} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix}^{\top} w_{ij}^* \right\}^{-1} \sum_{j=1}^{n} Y_j \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} w_{ij}^* \,, \tag{2}$$

with the weights $w_{ij}^* = K(|\Pi^* X_{ij}|^2/h^2)$, where h is some positive real number and Π^* is the orthogonal projector onto the EDR subspace $\mathcal{S}$. This choice of weights amounts to using infinite bandwidth in the directions lying in the orthogonal complement of the EDR subspace.

If only an estimator $\hat{A}$ of the orthogonal projector Π^* is available, a possible strategy is to replace Π^* by $\hat{A}$ in the definitions of the weights w_{ij}^*. This strategy is however too stringent, since it definitely discards the directions belonging to $\hat{\mathcal{S}}^{\perp}$. Being not sure that our information about the structure is exact, it is preferable to define the neighborhoods in a softer way. This is done by setting $w_{ij} = K(X_{ij}^{\top}(I+\rho^{-2}\hat{A})X_{ij}/h^2)$ and by redefining

$$\begin{pmatrix} \hat{f}(X_i) \\ \widehat{\nabla f}(X_i) \end{pmatrix} = \left\{ \sum_{j=1}^{n} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix}^{\top} w_{ij} \right\}^{-1} \sum_{j=1}^{n} Y_j \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} w_{ij} \,. \tag{3}$$

Here, ρ is a real number from the interval $[0,1]$ measuring the importance attributed to the estimator $\hat{A}$. If we are very confident in our estimator $\hat{A}$, we should choose ρ close to zero.

2.3 Recovering the EDR Subspace from an Estimator of ∇f

Suppose first that the values of the function ∇f at the points X_i are known. Then $\mathcal{S}$ is the linear subspace of $\mathbb{R}^d$ spanned by the vectors $\nabla f(X_i)$, $i=1,\dots,n$. For classifying the directions of $\mathbb{R}^d$ according to the variability of f in each direction and, as a by-product, identifying $\mathcal{S}$, the principal component analysis (PCA) can be used.

Recall that the PCA method is based on the orthogonal decomposition of the matrix $\mathscr{M} = n^{-1}\sum_{i=1}^{n} \nabla f(X_i)\nabla f(X_i)^{\top}$: $\mathscr{M} = O\Lambda O^T$ with an orthogonal matrix O and a diagonal matrix Λ with diagonal entries $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$. Clearly, for the multi-index model with m^*-indices, only the first m^* eigenvalues of $\mathscr{M}$ are positive. The first m^* eigenvectors of $\mathscr{M}$ (or, equivalently, the first m^* columns of the matrix O) define an orthonormal basis in the EDR subspace.

Let L be a positive integer. In Hristache et al. (2001a), a "truncated" matrix $\mathscr{M}_L$ is considered, which coincides with $\mathscr{M}$ if L equals n. Let $\{\psi_\ell, \ell = 1,\dots,L\}$ be a set of vectors of $\mathbb{R}^n$ satisfying the conditions $n^{-1}\sum_{i=1}^n \psi_{\ell,i}\psi_{\ell',i} = \delta_{\ell,\ell'}$ for every $\ell,\ell' \in \{1,\dots,L\}$, with $\delta_{\ell,\ell'}$ being the Kronecker symbol. Define

$$\beta_\ell = n^{-1}\sum_{i=1}^n \nabla f(X_i)\psi_{\ell,i} \tag{4}$$

and $\mathscr{M}_L = \sum_{\ell=1}^L \beta_\ell\beta_\ell^\top$. By the Bessel inequality, it holds $\mathscr{M}_L \preceq \mathscr{M}$. Here and in the sequel, for two symmetric matrices A and B, $A \preceq B$ means that $B-A$ is positive-semidefinite. Moreover, since $\mathscr{M}\mathscr{M}_L = \mathscr{M}_L\mathscr{M}$, any eigenvector of $\mathscr{M}$ is an eigenvector of $\mathscr{M}_L$. Finally, by the Parseval equality, $\mathscr{M}_L = \mathscr{M}$ if $L = n$.

The reason of considering the matrix $\mathscr{M}_L$ instead of $\mathscr{M}$ is that $\mathscr{M}_L$ can be estimated much better than $\mathscr{M}$. In fact, estimators of $\mathscr{M}$ have poor performance for samples of moderate size because of the sparsity of high dimensional data, ill-posedness of the gradient estimation and the non-linear dependence of $\mathscr{M}$ on ∇f. On the other hand, estimation of $\mathscr{M}_L$ reduces to the estimation of L linear functionals of ∇f and may be done with a better accuracy. The obvious limitation of this approach is that it recovers the EDR subspace entirely only if the rank of $\mathscr{M}_L$ coincides with the rank of $\mathscr{M}$, which is equal to m^*. To enhance our chances of seeing the condition $\operatorname{rank}(\mathscr{M}_L) = m^*$ fulfilled, we have to choose L sufficiently large. In practice, L is chosen of the same order as n.

In the case when only an estimator of ∇f is available, the above described method of recovering the EDR directions from an estimator of $\mathscr{M}_L$ has a risk of order $\sqrt{L/n}$ (Hristache et al., 2001a, Theorem 5.1). This fact advocates against using very large values of L. We desire nevertheless to use many linear combinations in order to increase our chances of capturing the whole EDR subspace. To this end, we modify the method of extracting the structural information from the estimators $\hat\beta_\ell$ of vectors β_ℓ.

Let $m \ge m^*$ be an integer. Observe that the estimator $\widetilde\Pi_m$ of the projector Π^* based on the PCA solves the following quadratic optimization problem:

$$\text{minimize } \sum_\ell \hat\beta_\ell^\top(I-\Pi)\hat\beta_\ell \qquad \text{subject to} \qquad \Pi^2 = \Pi, \quad \operatorname{tr}\Pi \le m, \tag{5}$$

where the minimization is carried over the set of all symmetric $(d\times d)$-matrices. The value m^* can be estimated by looking how many eigenvalues of $\widetilde\Pi_m$ are significant. Let $\mathcal{A}_m$ be the set of $(d\times d)$-matrices defined as follows:

$$\mathcal{A}_m = \{A : A = A^\top,\, 0 \preceq A \preceq I,\, \operatorname{tr}A \le m\}.$$

Define $\hat A_m$ as a minimizer of the maximum of the $\hat\beta_\ell^\top(I-A)\hat\beta_\ell$'s instead of their sum:

$$\hat A_m \in \operatorname*{arg\,min}_{A\in\mathcal{A}_m} \max_\ell \hat\beta_\ell^\top(I-A)\hat\beta_\ell. \tag{6}$$

This is a convex optimization problem that can be effectively solved even for a large d although a closed form solution is not known. It is noteworthy that a solution to (6) is not necessarily a projection matrix. In fact, the matrices from $\mathcal{A}_m$ are symmetric positive-semidefinite with eigenvalues between 0 and 1 and not just 0 or 1. This enlargement of the search space guarantees its convexity, which is needed for the algorithm to be tractable. Moreover, as we will show below, the incorporation of (6) in the structural adaptation yields an algorithm having good theoretical and empirical performance.

3. Theoretical Features of SAMM

Throughout this section the true dimension m^* of the EDR subspace is assumed to be known. Thus, we are given n observations $(Y_1,X_1),\dots,(Y_n,X_n)$ from the model

$$Y_i = f(X_i)+\varepsilon_i = g(\vartheta_1^\top X_i,\dots,\vartheta_{m^*}^\top X_i)+\varepsilon_i,$$

where $\varepsilon_1,\dots,\varepsilon_n$ are independent centered random variables. The vectors ϑ_j are assumed to form an orthonormal basis of the EDR subspace entailing thus the representation $\Pi^* = \sum_{j=1}^{m^*}\vartheta_j\vartheta_j^\top$. In what follows, we mainly consider deterministic design. Nevertheless, the results hold in the case of random design as well, provided that the errors are independent of $X_1,\dots,X_n$. Henceforth, without loss of generality we assume that $|X_i|\le 1$ for any $i=1,\dots,n$.

3.1 Description of the Algorithm

The structure-adaptive algorithm with maximum minimization consists of following steps.

a) Specify positive real numbers a_ρ, a_h, ρ_1 and h_1. Choose an integer L and select a set $\{\psi_\ell,\ \ell\le L\}$ of vectors from $\mathbb{R}^n$ verifying $|\psi_\ell|^2 = n$.

b) Set $k=1$ and $\hat A_0 = 0$.

c) Define the estimators $\widehat{\nabla f}_k(X_i)$ for $i=1,\dots,n$ by formula (3) with $w_{ij} = K\big(X_{ij}^\top(I+\rho_k^{-2}\hat A_{k-1})X_{ij}/h_k^2\big)$. Set

$$\hat\beta_{\ell,k} = \frac{1}{n}\sum_{i=1}^n \widehat{\nabla f}_k(X_i)\psi_{\ell,i},\qquad \ell=1,\dots,L,$$

where $\psi_{\ell,i}$ is the ith coordinate of ψ_ℓ.

d) Define the new value $\hat A_k$ by $\hat A_k \in \arg\min_{A\in\mathcal{A}_{m^*}}\max_\ell \hat\beta_{\ell,k}^\top(I-A)\hat\beta_{\ell,k}$.

e) Set $\rho_{k+1} = a_\rho\cdot\rho_k$, $h_{k+1} = a_h\cdot h_k$ and increase k by one.

f) Stop if $\rho_k<\rho_{\min}$ or $h_k>h_{\max}$, otherwise continue with the step c).

Let $k(n)$ be the total number of iterations. We denote by $\widehat\Pi_n$ the orthogonal projection onto the space spanned by the eigenvectors of $\hat A_{k(n)}$ corresponding to the m^* largest eigenvalues. The estimator of the EDR subspace is then the image of $\widehat\Pi_n$.

Both $\hat A_{k(n)}$ and $\widehat\Pi_n$ are estimators of the projector onto $\mathcal{S}$. Our theoretical results are stated for the estimator $\widehat\Pi_n$, but similar results are valid for $\hat A_{k(n)}$, too. The numerical simulations we made showed that these two estimators have comparable performances.

The described algorithm requires the specification of the parameters ρ_1, h_1, a_ρ and a_h, as well as the choice of the set of vectors $\{\psi_\ell\}$. In what follows we use the values

$$\begin{array}{ccc}\rho_1 = 1, & \rho_{\min} = n^{-1/(3\vee m^*)}, & a_\rho = e^{-1/2(3\vee m^*)},\\ h_1 = C_0 n^{-1/(4\vee d)}, & h_{\max} = 2\sqrt{d}, & a_h = e^{1/2(4\vee d)}.\end{array}$$

This choice of input parameters is up to some minor modifications the same as in Hristache et al. (2001b), Hristache et al. (2001a) and Samarov et al. (2005), and is based on the trade-off between

the bias and the variance of estimation. The constant C_0 will be chosen in a design-dependent manner taking into account the fact that the local neighborhoods used in (2) should contain enough design points to entail the consistency of the estimator. The choice of L and that of vectors ψ_ℓ will be discussed in Section 4.

3.2 Assumptions

Prior to stating rigorous theoretical results we need to introduce a set of assumptions. From now on, we use the notation I for the identity matrix of dimension d, $\|A\|^2$ for the largest eigenvalue of $A^\top A$ and $\|A\|_2$ for the Frobenius norm of A (square root of the sum of squares of elements of A).

We start with the smoothness assumption ensuring the adequacy of the local linear approximation of the regression function.

(A1) There exists a positive real C_g such that $|\nabla g(x)| \le C_g$ and $|g(x) - g(x') - (x-x')^\top \nabla g(x)| \le C_g |x-x'|^2$ for every $x, x' \in \mathbb{R}^{m^*}$.

Unlike the smoothness assumption, the assumptions on the identifiability of the model and the regularity of design are more involved and specific to our algorithm. The formal statements read as follows.

(A2) Let the vectors $\beta_\ell \in \mathbb{R}^d$ be defined by (4) and let $\mathcal{B}^* = \big\{\bar\beta = \sum_{\ell=1}^L c_\ell \beta_\ell : \sum_{\ell=1}^L |c_\ell| \le 1\big\}$. There exist vectors $\bar\beta_1, \dots, \bar\beta_{m^*} \in \mathcal{B}^*$ and constants $\mu_1, \dots, \mu_{m^*}$ such that

$$\Pi^* \preceq \sum_{k=1}^{m^*} \mu_k \bar\beta_k \bar\beta_k^\top. \tag{7}$$

We denote $\mu^* = \mu_1 + \ldots + \mu_{m^*}$.

Remark 1 *Assumption (A2) implies that the subspace $\mathcal{S} = \mathrm{Im}(\Pi^*)$ is the smallest DR subspace, therefore it is the EDR subspace. Indeed, for any DR subspace $\mathcal{S}'$, the gradient $\nabla f(X_i)$ belongs to $\mathcal{S}'$ for every i. Therefore $\beta_\ell \in \mathcal{S}'$ for every $\ell \le L$ and $\mathcal{B}^* \subset \mathcal{S}'$. Thus, for every β° from the orthogonal complement $\mathcal{S}'^\perp$, it holds $|\Pi^* \beta^\circ|^2 \le \sum_k \mu_k |\bar\beta_k^\top \beta^\circ|^2 = 0$. Therefore $\mathcal{S}'^\perp \subset \mathcal{S}^\perp$ implying thus the inclusion $\mathcal{S} \subset \mathcal{S}'$.*

Lemma 2 *If the family $\{\psi_\ell\}$ spans $\mathbb{R}^n$, then assumption (A2) is always satisfied with some μ_k (that may depend on n).*

Proof Set $\Psi = (\psi_1, \dots, \psi_L) \in \mathbb{R}^{n \times L}$, $\nabla f = (\nabla f(X_1), \dots, \nabla f(X_n)) \in \mathbb{R}^{d \times n}$ and write the $d \times L$ matrix $B = (\beta_1, \dots, \beta_L)$ in the form $\nabla f \cdot \Psi$. Recall that if M_1, M_2 are two matrices such that $M_1 \cdot M_2$ is well defined and the rank of M_2 coincides with the number of lines in M_2, then $\mathrm{rank}(M_1 \cdot M_2) = \mathrm{rank}(M_1)$. This implies that $\mathrm{rank}(B) = m^*$ provided that $\mathrm{rank}(\Psi) = n$, which amounts to $\mathrm{span}(\{\psi_\ell\}) = \mathbb{R}^n$.

Let now $\tilde\beta_1, \dots, \tilde\beta_{m^*}$ be a linearly independent subfamily of $\{\beta_\ell, \ell \le L\}$. Then the m^*th largest eigenvalue $\lambda_{m^*}(\tilde{\mathscr{M}})$ of the matrix $\tilde{\mathscr{M}} = \sum_{k=1}^{m^*} \tilde\beta_k \tilde\beta_k^\top$ is strictly positive. Moreover, if $v_1, \dots, v_{m^*}$ are the eigenvectors of $\tilde{\mathscr{M}}$ corresponding to the eigenvalues $\lambda_1(\tilde{\mathscr{M}}) \ge \ldots \ge \lambda_{m^*}(\tilde{\mathscr{M}}) > 0$, then

$$\Pi^* = \sum_{k=1}^{m^*} v_k v_k^\top \preceq \frac{1}{\lambda_{m^*}(\tilde{\mathscr{M}})} \sum_{k=1}^{m^*} \lambda_k(\tilde{\mathscr{M}}) v_k v_k^\top = \lambda_{m^*}(\tilde{\mathscr{M}})^{-1} \tilde{\mathscr{M}} = \lambda_{m^*}(\tilde{\mathscr{M}})^{-1} \sum_{k=1}^{m^*} \tilde\beta_k \tilde\beta_k^\top.$$

Hence, inequality (7) is fulfilled with $\mu_k = \lambda_{m^*}(\tilde{\mathscr{M}})^{-1}$ for every $k = 1, \dots, m^*$. ■

These arguments show that the identifiability assumption (A2) is fairly weak. In fact, since we always choose $\{\psi_\ell\}$ so that $\mathrm{span}(\{\psi_\ell\}) = \mathbb{R}^n$, (A2) amounts to requiring that the value μ^* remains bounded when n increases. This assumption is much weaker than the coverage assumption under which the consistency of the inverse regression based methods is proved.

Let us proceed with the assumption on the design regularity. Define $P_1^* = I$ and $P_k^* = (I + \rho_k^{-2}\Pi^*)^{-1/2}$ for every $k \geq 2$. Next, set $Z_{ij}^{(k)} = (h_k P_k^*)^{-1} X_{ij}$ and for any $d \times d$ matrix U put $w_{ij}^{(k)}(U) = K\big((Z_{ij}^{(k)})^\top U Z_{ij}^{(k)}\big)$, $\overline{w}_{ij}^{(k)}(U) = K'\big((Z_{ij}^{(k)})^\top U Z_{ij}^{(k)}\big)$, $N_i^{(k)}(U) = \sum_j w_{ij}^{(k)}(U)$ and

$$V_i^{(k)}(U) = \sum_{j=1}^n \begin{pmatrix} 1 \\ Z_{ij}^{(k)} \end{pmatrix} \begin{pmatrix} 1 \\ Z_{ij}^{(k)} \end{pmatrix}^\top w_{ij}^{(k)}(U).$$

(A3) For some positive constants $C_V, C_K, C_{K'}, C_w$ and for some $\alpha \in]0, 1/2]$, the inequalities

$$\begin{aligned}
&\|V_i^{(k)}(U)^{-1}\| N_i^{(k)}(U) \leq C_V, \qquad i = 1, \dots, n, \\
&\sum_{i=1}^n w_{ij}^{(k)}(U)/N_i^{(k)}(U) \leq C_K, \qquad j = 1, \dots, n, \\
&\sum_{i=1}^n |\overline{w}_{ij}^{(k)}(U)|/N_i^{(k)}(U) \leq C_{K'}, \qquad j = 1, \dots, n, \\
&\sum_{j=1}^n |\overline{w}_{ij}^{(k)}(U)|/N_i^{(k)}(U) \leq C_w \qquad i = 1, \dots, n,
\end{aligned}$$

hold for every $k \leq k(n)$ and for every $d \times d$ matrix U verifying $\|U - I\|_2 \leq \alpha$.

Remark 3 *Note that in (A3) we implicitly assumed that the matrices $V_i^{(k)}$ are invertible, which may be true only if any neighborhood $E^{(k)}(X_i) = \{x : |(I + \rho_k^{-2}\Pi^*)^{-1/2}(X_i - x)| \leq h_k\}$ contains at least d design points different from X_i. The parameters h_1, ρ_1, a_ρ and a_h are chosen so that the volume of ellipsoids $E^{(k)}(X_i)$ is a non-decreasing function of k and $Vol(E^{(1)}(X_i)) = C_0/n$. Therefore, from theoretical point of view, if the design is random with positive density on $[0,1]^d$, it is easy to check that for a properly chosen constant C_0, assumption (A3) is satisfied with a probability close to one. In applications, we define h_1 as the smallest real such that $\min_{i=1,\dots,n} \# E^{(1)}(X_i) = d + 1$ and add to the matrix*

$$\sum_{j=1}^n \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix} \begin{pmatrix} 1 \\ X_{ij} \end{pmatrix}^\top w_{ij},$$

involved in the definition (3), a small full-rank matrix to be sure that the resulting matrix is invertible, see Section 4. This observation also implies that the SAMM procedure can not be applied in the case where the sample size n is smaller than or equal to the dimension d of predictors.

(A4) The errors $\{\varepsilon_i, i \leq n\}$ are centered Gaussian with variance σ^2.

3.3 Risk Bounds for the Projection Matrix Estimation

In this section, we present the main result of the paper assessing the quality of the estimator $\widehat{\Pi}_n$ of the projection matrix Π^* in the asymptotics of large samples. To this end, we assume that the kernel K used in (3) is chosen to be continuous, positive and vanishing outside the interval $[0,1]$. The vectors ψ_ℓ are assumed to verify

$$\max_{\ell=1,\dots,L}\max_{i=1,\dots,n}|\psi_{\ell,i}|<\bar{\psi}, \tag{8}$$

for some constant $\bar{\psi}$ independent of n. In the sequel, we denote by $C,C_1,\dots$ some constants depending only on $m^*,\mu^*,C_g,C_V,C_K,C_{K'},C_w$ and $\bar{\psi}$.

Theorem 4 *Let assumptions **(A1)-(A4)** be fulfilled. There exists $C>0$ such that for any $z\in]0,2\sqrt{\log(nL)}]$ and for sufficiently large values of n, it holds*

$$\mathbf{P}\left(\sqrt{\mathrm{tr}(I-\widehat{\Pi}_n)\Pi^*}>Cn^{-\frac{2}{3\vee m^*}}t_n^2+\frac{2\sqrt{\mu^*}zc_0\sigma}{\sqrt{n(1-\zeta_n)}}\right)\le Lze^{-\frac{z^2-1}{2}}+\frac{3k(n)-5}{n},$$

where $c_0=\bar{\psi}\sqrt{dC_KC_V}$, $t_n=O(\sqrt{\log(Ln)})$ and $\zeta_n=O(t_n n^{-\frac{1}{6\vee m^*}})$.

Corollary 5 *Under the assumptions of Theorem 4, for sufficiently large n, it holds*

$$\mathbf{P}\left(\|\widehat{\Pi}_n-\Pi^*\|_2>Cn^{-\frac{2}{3\vee m^*}}t_n^2+\frac{2\sqrt{2\mu^*}zc_0\sigma}{\sqrt{n(1-\zeta_n)}}\right)\le Lze^{-\frac{z^2-1}{2}}+\frac{3k(n)-5}{n}$$

$$\mathbf{E}(\|\widehat{\Pi}_n-\Pi^*\|_2)\le C\left(n^{-\frac{2}{3\vee m^*}}t_n^2+\frac{\sqrt{\log nL}}{\sqrt{n}}\right)+\frac{\sqrt{2m^*}(3k(n)-5)}{n}.$$

Proof Easy algebra yields

$$\begin{aligned}\|\widehat{\Pi}_n-\Pi^*\|_2^2=\mathrm{tr}(\widehat{\Pi}_n-\Pi^*)^2&=\mathrm{tr}\,\widehat{\Pi}_n^2-2\,\mathrm{tr}\,\widehat{\Pi}_n\Pi^*+\mathrm{tr}\,\Pi^*\\&\le\mathrm{tr}\,\widehat{\Pi}_n+m^*-2\,\mathrm{tr}\,\widehat{\Pi}_n\Pi^*\le 2m^*-2\,\mathrm{tr}\,\widehat{\Pi}_n\Pi^*.\end{aligned}$$

The equality $\mathrm{tr}\,\Pi^*=m^*$ and the linearity of the trace operator complete the proof of the first inequality. The second inequality can be derived from the first one by standard arguments in view of the inequality $\|\widehat{\Pi}_n-\Pi^*\|_2^2\le 2m^*$. ■

According to these results, for $m^*\le 4$, the estimator of the orthogonal projector onto $\mathcal{S}$ provided by the SAMM procedure is $\sqrt{n}$-consistent up to a logarithmic factor. This rate of convergence is known to be optimal for a broad class of semiparametric problems, see Bickel et al. (1998) for a detailed account on the subject.

Remark 6 *The inspection of the proof of Theorem 4 shows that the factor t_n^2 multiplying the "bias" term $n^{-2/(3\vee m^*)}$ disappears when $m^*>3$.*

Remark 7 *The same rate of convergence remains valid in the case when the errors are not necessarily identically distributed Gaussian random variables, but have a bounded exponential moment (uniformly in n). This can be proved along the lines of Proposition 14, see Section 5.*

3.4 Risk Bound for the Estimator of a Basis of the EDR Subspace

The main result of this paper stated in the preceding subsection provides a risk bound for the estimator $\widehat{\Pi}_n$ of Π^*, the orthogonal projector onto $\mathcal{S}$. As a by-product of this result, we show in this section that a similar risk bound holds also for the estimator of an orthonormal basis of $\mathcal{S}$. This means that for an arbitrarily chosen orthonormal basis of the estimated EDR subspace $\hat{\mathcal{S}} = \mathrm{Im}(\widehat{\Pi}_n)$, there is an orthonormal basis of the true EDR subspace $\mathcal{S}$ such that the matrices built from these bases are close in Frobenius norm with a probability tending to one.

Proposition 8 *Let the assumptions of Theorem 4 be fulfilled. For any orthonormal basis $\hat{\vartheta}_1, \ldots, \hat{\vartheta}_{m^*}$ of the estimated EDR subspace $\hat{\mathcal{S}} = \mathrm{Im}(\widehat{\Pi}_n)$ there exists an orthonormal basis $\vartheta_1, \ldots, \vartheta_{m^*}$ of the true EDR subspace $\mathcal{S} = \mathrm{Im}(\Pi^*)$ such that, for sufficiently large n, it holds*

$$\mathbf{P}\left(\|\widehat{\Theta}_n - \Theta\|_2 > C n^{-\frac{2}{3\vee m^*}} t_n^2 + \frac{2(\sqrt{m^*}+1)\sqrt{2\mu^*} z c_0 \sigma}{\sqrt{n(1-\zeta_n)}} \right) \leq L z e^{-\frac{z^2-1}{2}} + \frac{3k(n)-5}{n} ,$$

where $\widehat{\Theta}_n$ (resp. Θ) is the $d \times m^*$ matrix whose jth column is $\hat{\vartheta}_j$ (resp. ϑ_j).

Proof Using the singular value decomposition, we write $\Pi^* \widehat{\Theta}_n = U \Lambda V^\top$, where U and V are orthogonal matrices and Λ is a diagonal matrix. Let us denote by λ_j, u_j, v_j respectively the jth diagonal entry of Λ, the jth column of U and the jth column of V. Since $\Pi^* \widehat{\Theta}_n v_j = \lambda_j u_j$, we have $\lambda_j = |\lambda_j u_j| = |\Pi^* \widehat{\Theta}_n v_j| \leq 1$. On the other hand,

$$\lambda_j = |\Pi^* \widehat{\Theta}_n v_j| \geq |\widehat{\Theta}_n v_j| - |(\widehat{\Pi}_n - \Pi^*)\widehat{\Theta}_n v_j| \geq 1 - \|\widehat{\Pi}_n - \Pi^*\|,$$

where we used the fact that $|\widehat{\Theta}_n v_j|^2 = v_j^\top \widehat{\Theta}_n^\top \widehat{\Theta}_n v_j = v_j^\top v_j = 1$. Let us define the matrix Θ as follows: $\Theta = U I_{d \times m^*} V^\top$, where $I_{d \times m^*}$ is the $d \times m^*$ diagonal matrix with all diagonal entries equal to one. One easily checks that Θ is orthogonal, that is $\Theta^\top \Theta = I_{m^*}$. Moreover, we have $\Theta = \Pi^* \hat{\Theta}_n V \Lambda^- V^\top$, where Λ^- is the $m^* \times m^*$ diagonal matrix having λ_j^{-1} as jth diagonal entry. Note that if the norm of $\hat{\Pi}_n - \Pi^*$ is less than 1, the eigenvalues λ_j are strictly positive. In this case, Λ^- is well defined and we obviously have $\Pi^* \Theta = \Theta$. Thus the columns of Θ form an orthonormal basis of $\mathrm{Im}(\Pi^*)$. Furthermore, we have

$$\begin{aligned}
\|\widehat{\Theta}_n - \Theta\|_2 &\leq \|\Theta - \Pi^* \widehat{\Theta}_n\|_2 + \|(\Pi^* - \widehat{\Pi}_n)\Theta\|_2 \\
&\leq \|U(I_{d \times m^*} - \Lambda) V^\top\|_2 + \|\Pi^* - \widehat{\Pi}_n\|_2 \\
&\leq (\sum_{j=1}^{m^*} (\lambda_j - 1)^2)^{1/2} + \|\Pi^* - \widehat{\Pi}_n\|_2 \\
&\leq (\sqrt{m^*} + 1)\|\Pi^* - \widehat{\Pi}_n\|_2,
\end{aligned}$$

provided that $\|\Pi^* - \widehat{\Pi}_n\| < 1$. This implies that for every $\mathsf{d} \in (0,1)$ the event $\{\|\Pi^* - \widehat{\Pi}_n\|_2 \leq \mathsf{d}\}$ is included in $\{\|\widehat{\Theta}_n - \Theta\|_2 \leq (\sqrt{m^*}+1)\mathsf{d}\}$. By virtue of this inclusion, the assertion of the proposition follows from Corollary 5. ■

4. Simulation Results

The aim of this section is to demonstrate on several examples how the performance of the algorithm SAMM depends on the sample size n, the dimension d and the noise level σ. We also show that our procedure can be successfully applied in autoregressive models. Many unreported results show that in most situations the performance of SAMM is comparable to the performance of SA approach based on PCA and to that of MAVE. A thorough comparison of the numerical virtues of these methods being out of scope of this paper, we simply show on some examples that SAMM may substantially outperform MAVE in the case of large "bias". Our results also show that SAMM and MAVE provide more accurate estimates of the EDR subspace than inverse regression based methods : inverse regression based on Minimum Discrepancy Approach (MDA) introduced in Cook and Ni (2005) and Sliced Average Variance Estimation (SAVE) of Cook and Weisberg (1991). In all simulations for inverse regression based methods, the number of slices is chosen to minimise the risk.

The computer code of the procedure SAMM is distributed freely, it can be downloaded from `http://code.google.com/p/samm07/`. It requires the MATLAB packages SDPT3 and YALMIP. We are grateful to Professor Yingcun Xia for making the computer code of MAVE available to us.

To obtain higher stability of the algorithm, we preliminarily standardize the response Y and the predictors $X^{(j)}$. More precisely, we deal with $\tilde{Y}_i = Y_i/\sigma_Y$ and $\tilde{X} = \mathrm{diag}(\Sigma_X)^{-1/2}X$, where σ_Y^2 is the empirical variance of Y, Σ_X is the empirical covariance matrix of X and $\mathrm{diag}(\Sigma_X)$ is the $d \times d$ matrix obtained from Σ_X by replacing the off-diagonal elements by zero. To preserve consistency, we set $\tilde{\beta}_{\ell,k(n)} = \mathrm{diag}(\Sigma_X)^{-1/2}\hat{\beta}_{\ell,k(n)}$, where $\hat{\beta}_{\ell,k(n)}$ is the last-step estimate of β_ℓ, and define $\widehat{\Pi}_{k(n)}$ as the solution to (6) with $\hat{\beta}_\ell$ replaced by $\tilde{\beta}_{\ell,k(n)}$. Furthermore, we add the small full-rank matrix I_{d+1}/n to $\sum_{j=1}^n \binom{1}{X_{ij}}\binom{1}{X_{ij}}^\top w_{ij}$ in (3).

In all examples presented below the number of replications is $N = 250$; for each replication, a new sample of the design and the error vector $(\varepsilon_1, \ldots, \varepsilon_n)$ has been generated at random. The mean loss $\overline{\mathrm{er}_N} = \frac{1}{N}\sum_j \mathrm{er}_j$ and the standard deviation $\sqrt{\frac{1}{N}\sum_j (\mathrm{er}_j - \overline{\mathrm{er}_N})^2}$ are reported, where $\mathrm{er}_j = \|\widehat{\Pi}^{(j)} - \Pi^*\|$ with $\widehat{\Pi}^{(j)}$ being the estimator of Π^* for jth replication.

4.1 Choice of $\{\psi_\ell, \ell \le L\}$

The set $\{\psi_\ell\}$ plays an essential role in the algorithm. The optimal choice of this set is an important issue that needs further investigation. We content ourselves with giving one particular choice which agrees with theory and leads to nice empirical results.

Let $\mathfrak{S}_j$, $j \le d$, be the permutation of the set $\{1, \ldots, n\}$ satisfying $X^{(j)}_{\mathfrak{S}_j(1)} \le \ldots \le X^{(j)}_{\mathfrak{S}_j(n)}$. Let $\mathfrak{S}_j^{-1}$ be the inverse of $\mathfrak{S}_j$, that is, $\mathfrak{S}_j(\mathfrak{S}_j^{-1}(k)) = k$ for every $k = 1, \ldots, n$. Define $\{\psi_\ell\}$ as the set of vectors

$$\left\{ \begin{pmatrix} \cos\big(\frac{2\pi(k-1)\mathfrak{S}_j^{-1}(1)}{n}\big), \ldots, \cos\big(\frac{2\pi(k-1)\mathfrak{S}_j^{-1}(n)}{n}\big) \end{pmatrix}^\top \\ \begin{pmatrix} \sin\big(\frac{2\pi k\mathfrak{S}_j^{-1}(1)}{n}\big), \ldots, \sin\big(\frac{2\pi k\mathfrak{S}_j^{-1}(n)}{n}\big) \end{pmatrix}^\top , k \le [n/2],\ j \le d \right\}$$

normalized to satisfy $\sum_{i=1}^n \psi_{\ell,i}^2 = n$ for every ℓ. It is easily seen that these vectors satisfy conditions (8) and $\mathrm{span}(\{\psi_\ell\}) = \mathbb{R}^n$, so the conclusion of Lemma 2 holds. Above, $[n/2]$ is the integer part of $n/2$ and k and j are positive integers.

The idea behind the above described choice of vectors ψ_ℓ is the following: if the design is uniformly distributed in $[0,1]^d$ and $H(x)$ is a function $\mathbb{R}^d \to \mathbb{R}$ depending only on one coordinate of x, the projections of the vector $g = (H(X_1),\dots,H(X_n))^\top$ on some of directions ψ_ℓ are nearly equal to the Fourier coefficients of H. Indeed, for n odd and for every fixed j, the vectors $\{e_{k,j} = \big(\phi_k(\mathfrak{S}_j^{-1}(1)/n), \phi_k(\mathfrak{S}_j^{-1}(2)/n),\dots,\phi_k(\mathfrak{S}_j^{-1}(n)/n)\big)^\top; 1 \le k \le n\}$ with $\{\phi_k\}$ being the trigonometric basis (that is $\phi_{2p}(x) = \sqrt{2}\sin(2\pi px)$ and $\phi_{2p+1}(x) = \sqrt{2}\cos(2\pi px)$ for every $p \in \mathbb{N}$) form an orthonormal basis of $\mathbb{R}^n$. Therefore, for any function H from $\mathbb{R}^d$ to $\mathbb{R}$, which depends exclusively on the j^{th} coordinate $x^{(j)}$ of x, one has $g^\top e_{kj} = \sum_{i=1}^n H_0(X_i^{(j)})\phi_k(\mathfrak{S}_j^{-1}(i)/n) = \sum_{i=1}^n H_0(X^{(j)}_{\mathfrak{S}_j(i)})\phi_k(i/n)$ for some function $H_0 : \mathbb{R} \to \mathbb{R}$. Since for a sample $X_1^{(j)},\dots,X_n^{(j)}$ drawn from uniform distribution in $[0,1]$ the order statistics are nearly equal to i/n, we get $\frac{1}{n} g^\top e_{kj} \approx \frac{1}{n}\sum_{i=1}^n H_0(i/n)\phi_k(i/n) \approx \langle H_0, \phi_k\rangle_{L^2[0,1]}$. Note that although this explanation is valid only for uniform design and a function H depending only on one coordinate, empirical results show that this choice leads to satisfactory results in more general situations.

4.2 Example 1 (Single-index)

We set $d = 5$ and $f(x) = g(\vartheta^\top x)$ with

$$g(t) = 4|t|^{1/2}\sin^2(\pi t), \quad \text{and} \quad \vartheta = (1/\sqrt{5}, 2/\sqrt{5}, 0, 0, 0)^\top \in \mathbb{R}^5.$$

We ran SAMM, MAVE, MDA and SAVE procedures on the data generated by the model

$$Y_i = f(X_i) + 0.5\cdot\varepsilon_i,$$

where the design X is such that the coordinates $(X_i^{(j)}, j \le 5, i \le n)$ are i.i.d. uniform in $[-1,1]$, and the errors ε_i are i.i.d. standard Gaussian independent of the design.

Table 1 contains the average loss for different values of the sample size n for the first step estimator by SAMM, the final estimator provided by SAMM and the estimators based on MAVE, MDA and SAVE. The first observation is that inverse regression based methods are not consistent in this case. We plot in Figure 1 the average loss normalized by the square root of the sample size n versus n. It is clearly seen that the iterative procedure improves considerably the quality of estimation and that the final estimator provided by SAMM is $\sqrt{n}$-consistent. In this example, MAVE method often fails to recover the EDR subspace. However, the number of failures decreases very rapidly with increasing n. This is the reason why the curve corresponding to MAVE in Figure 1 decreases with a strong slope.

4.3 Example 2 (Double-index)

For $d \ge 2$ we set $f(x) = g(\vartheta^\top x)$ with

$$g(x) = (x_1 - x_2^3)(x_1^3 + x_2);$$

and $\vartheta_1 = (1,0,\dots,0) \in \mathbb{R}^d$, $\vartheta_2 = (0,1,\dots,0) \in \mathbb{R}^d$. We ran SAMM, MAVE, MDA and SAVE procedures on the data generated by the model

$$Y_i = f(X_i) + 0.1\cdot\varepsilon_i, \qquad i = 1,\dots,300,$$

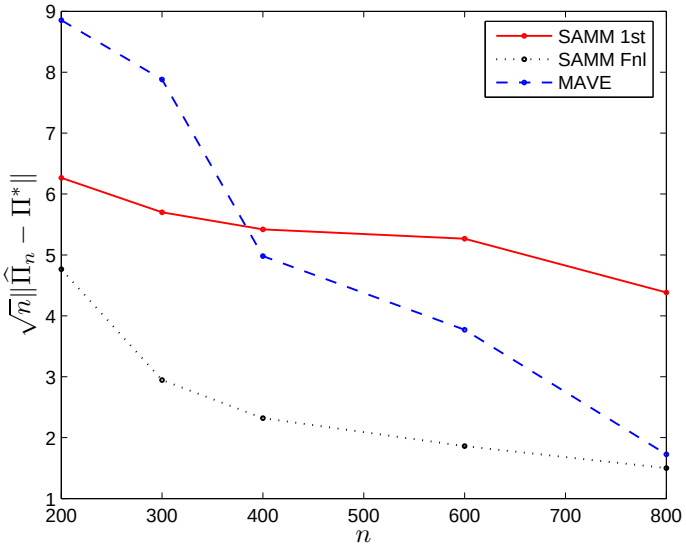


Figure 1: Average loss multiplied by $\sqrt{n}$ versus n for the first step (solid line) and the final (dotted line) estimators provided by SAMM and for the estimator by MAVE (dashed line) in Example 1.

n	200	300	400	600	800
SAMM, 1st	0.443	0.329	0.271	0.215	0.155
	(.211)	(.120)	(.115)	(.095)	(.079)
SAMM, Fnl	0.337	0.170	0.116	0.076	0.053
	(.273)	(.147)	(.104)	(.054)	(.031)
MAVE	0.626	0.455	0.249	0.154	0.061
	(.363)	(.408)	(.342)	(.290)	(.161)
MDA	0.882	0.885	0.890	0.885	0.882
	(.144)	(.141)	(.130)	(.142)	(.148)
SAVE	0.857	0.847	0.832	0.818	0.782
	(.145)	(.144)	(.154)	(.168)	(.169)

Table 1: Average loss $\|\widehat{\Pi} - \Pi^*\|$ of the estimators obtained by SAMM, MAVE, MDA and SAVE procedures in Example 1. The standard deviation is given in parentheses.

where the design X is such that the coordinates $(X_i^{(j)}, j \le d, i \le n)$ are i.i.d. uniform in $[-40,40]$, and the errors ε_i are i.i.d. standard Gaussian independent of the design. The results of simulations for different values of d are reported in Table 2.

As expected, we found that (cf. Figure 2) the quality of SAMM, as well as the quality of SAVE, deteriorated linearly in d as d increased. This agrees with our theoretical results. It should be noted that in this case MAVE and MDA fail to find the EDR subspace.

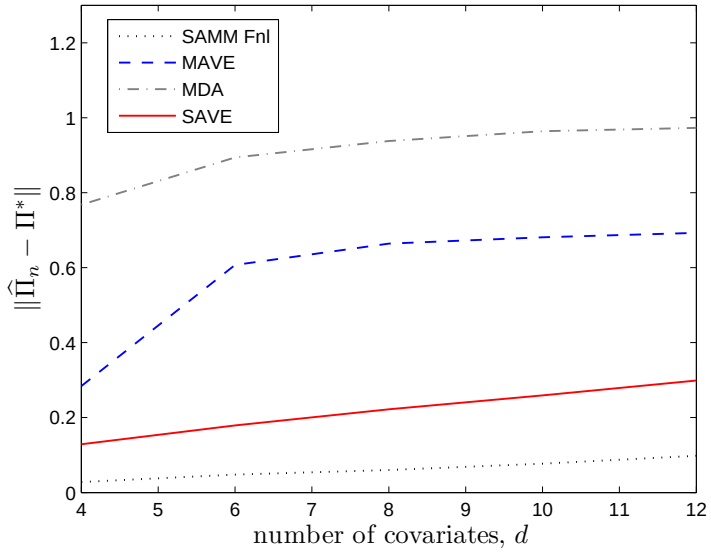


Figure 2: Average loss versus d for the estimators provided by SAMM (dotted line), by MAVE (dashed line), by MDA (dash-dot line) and by SAVE (solid line) in Example 2.

d	4	6	8	10	12
SAMM 1st	0.154	0.242	0.296	0.365	0.421
	(.063)	(.081)	(.071)	(.087)	(.095)
SAMM, Fnl	0.028	0.048	0.060	0.077	0.098
	(.011)	(.020)	(.021)	(.026)	(.037)
MAVE	0.284	0.607	0.664	0.681	0.693
	(.147)	(.073)	(.052)	(.054)	(.044)
MDA	0.768	0.894	0.938	0.964	0.973
	(.232)	(.142)	(.095)	(.062)	(.049)
SAVE	0.129	0.179	0.222	0.259	0.299
	(.048)	(.047)	(.050)	(.058)	(.071)

Table 2: Average loss $\|\widehat{\Pi} - \Pi^*\|$ of the estimators obtained by SAMM, MAVE and MDA procedures in Example 2. The standard deviation is given in parentheses.

4.4 Example 3

For $d = 5$ we set $f(x) = g(\vartheta^\top x)$ with

$$g(x) = (1+x_1)(1+x_2)(1+x_3)$$

and $\vartheta_1 = (1,0,0,0,0)$, $\vartheta_2 = (0,1,0,0,0)$, $\vartheta_3 = (0,0,1,0,0)$. We ran SAMM, MAVE, MDA and SAVE procedures on the data generated by the model

$$Y_i = f(X_i) + \sigma \cdot \varepsilon_i, \qquad i = 1, \ldots, 250,$$

σ	200	150	100	50	25	10
SAMM 1st	0.227	0.177	0.141	0.119	0.113	0.106
	(.092)	(.075)	(.055)	(.051)	(.048)	(.043)
SAMM, Fnl	0.125	0.084	0.057	0.039	0.034	0.030
	(.076)	(.037)	(.026)	(.019)	(.021)	(.018)
MAVE	0.103	0.087	0.073	0.062	0.063	0.059
	(.041)	(.035)	(.027)	(.023)	(.024)	(.023)
MDA	0.854	0.850	0.867	0.862	0.858	0.873
	(.167)	(.173)	(.157)	(.159)	(.171)	(.159)
SAVE	0.510	0.511	0.496	0.505	0.496	0.490
	(.208)	(.204)	(.207)	(.197)	(.196)	(.199)

Table 3: Average loss $\|\widehat{\Pi} - \Pi^*\|$ of the estimators obtained by SAMM and MAVE procedures in Example 3. The standard deviation is given in parentheses.

where the design X is such that the coordinates $(X_i^{(j)}, j \le d, i \le n)$ are i.i.d. uniform in $[0,20]$, and the errors ε_i are i.i.d. standard Gaussian independent of the design.

Figure 3 shows that the qualities of both SAMM and MAVE deteriorate linearly in σ, when σ increases. These results also demonstrate that, thanks to an efficient bias reduction, the SAMM procedure outperforms MAVE when stochastic error is small, whereas MAVE works better than SAMM in the case of dominating stochastic error (that is when σ is large).

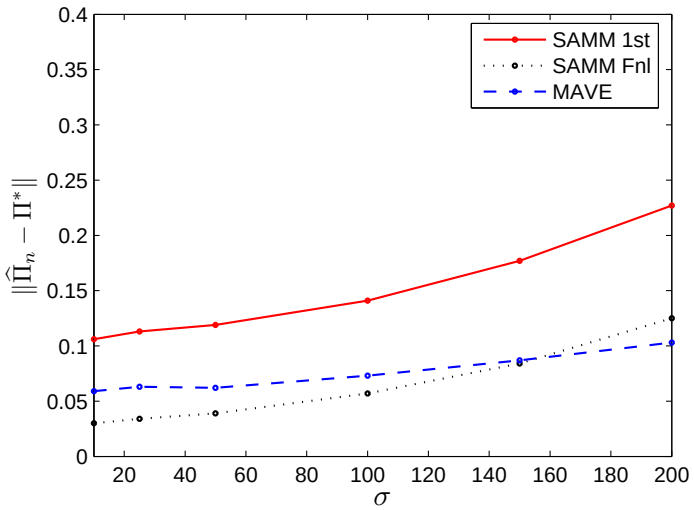


Figure 3: Average loss versus σ for the first step (solid line) and the final (dotted line) estimators provided by SAMM and for the estimator based on MAVE (dashed line) in Example 3.

4.5 Example 4 (Time Series)

Let now $T_1,\dots,T_{n+6}$ be generated by the autoregressive model

$$T_{i+6} = f(T_{i+5},T_{i+4},T_{i+3},T_{i+2},T_{i+1},T_i) + 0.2\cdot\varepsilon_i, \qquad i=1,\dots,n,$$

with initial variables $T_1,\dots,T_6$ being independent standard normal independent of the innovations ε_i, which are i.i.d. standard normal as well. Let now $f(x) = g(\vartheta^\top x)$ with

$$g(x) = -1 + 0.6x_1 - \cos(0.5\pi x_2) + e^{-x_3^2},$$

and

$$\begin{aligned}\vartheta_1 &= (1,0,0,2,0,0)/\sqrt{5},\\ \vartheta_2 &= (0,0,1,0,0,2)/\sqrt{5},\\ \vartheta_3 &= (-2,2,-2,1,-1,1)/\sqrt{15}.\end{aligned}$$

We ran SAMM and MAVE procedures on the data (X_i,Y_i), $i=1,\dots,250$, where $Y_i = T_{i+6}$ and $X_i = (T_i,\dots,T_{i+5})^\top$. The results of simulations reported in Table 4 show that the qualities of SAMM and MAVE are comparable, with SAMM being slightly better. SAVE is better than MDA, but both of them are far less accurate than SAMM and MAVE.

n	300	400	500	600
SAMM, 1st	0.391	0.351	0.334	0.293
	(.172)	(.161)	(.137)	(.132)
SAMM, Fnl	0.220	0.186	0.174	0.146
	(.119)	(.123)	(.102)	(.089)
MAVE	0.268	0.231	0.209	0.182
	(.209)	(.170)	(.159)	(.122)
MDA	0.914	0.915	0.913	0.912
	(.115)	(.107)	(.119)	(.119)
SAVE	0.617	0.515	0.428	0.369
	(.200)	(.184)	(.151)	(.138)

Table 4: Average loss $\|\widehat{\Pi} - \Pi^*\|$ of the estimators obtained by SAMM, MAVE, MDA and SAVE procedures in Example 4. The standard deviation is given in parentheses.

5. Proofs

Since the proof of the main result is carried out in several steps, we give a short road map for guiding the reader throughout the proof. The main idea is to evaluate the accuracy of the first step estimators of β_ℓ and, given the accuracy of the estimator at the step k, evaluate the accuracy of the estimators at the step $k+1$. This is done in Subsections 5.2 and 5.1. These results are based on a maximal inequality proved in Subsection 5.4 and on some properties of the solution to (6) proved in Subsection 5.5. The proof of Theorem 4 is presented in Subsection 5.3, while some technical lemmas are postponed to Subsection 5.6.

5.1 One Step Improvement

Let $\{\delta_k\}$ be a sequence of positive numbers to be chosen later and let $\mathscr{P}_k = \big\{A \in \mathcal{A}_{m^*} : \operatorname{tr}(I-A)\Pi^* \le \delta_k^2\big\}$. Recall that we use the following notation:

$$P_k^* = (I+\rho_k^{-2}\Pi^*)^{-1/2}, \quad Z_{ij}^{(k)} = (h_k P_k^*)^{-1} X_{ij}, \quad w_{ij}^{(k)}(U) = K\big((Z_{ij}^{(k)})^\top U Z_{ij}^{(k)}\big)$$
$$N_i^{(k)}(U) = \sum_j w_{ij}^{(k)}(U), \qquad V_i^{(k)}(U) = \sum_{j=1}^n \begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}^\top w_{ij}^{(k)}(U),$$

where U is a $d\times d$ symmetric positive-semidefinite matrix. Let us define $S_k = (I+\rho_k^{-2}\hat{A}_{k-1})^{1/2}$ and $U_k = P_k^* S_k^2 P_k^*$.

One easily checks that the estimator $\widehat{\nabla f}_k(X_i)$ is given by

$$\begin{pmatrix}\hat{f}_k(X_i)\\ \widehat{\nabla f}_k(X_i)\end{pmatrix} = \Big\{\sum_{j=1}^n \begin{pmatrix}1\\ X_{ij}\end{pmatrix}\begin{pmatrix}1\\ X_{ij}\end{pmatrix}^\top w_{ij}^{(k)}(U_k)\Big\}^{-1} \sum_{j=1}^n Y_j \begin{pmatrix}1\\ X_{ij}\end{pmatrix} w_{ij}^{(k)}(U_k),$$

Simple algebra yields

$$\begin{pmatrix}h_k^{-1}\hat{f}_k(X_i)\\ P_k^*\widehat{\nabla f}_k(X_i)\end{pmatrix} = h_k^{-1} V_i^{(k)}(U_k)^{-1} \sum_{j=1}^n Y_j \begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix} w_{ij}^{(k)}(U_k).$$

In order to study the behavior of $\widehat{\nabla f}_k$, we will proceed in a first step as if U_k were deterministic. For this reason, the notation

$$\begin{pmatrix}h_k^{-1}\overline{f}_k(X_i)\\ P_k^*\overline{\nabla f}_k(X_i)\end{pmatrix} = h_k^{-1} V_i^{(k)}(U_k)^{-1} \sum_{j=1}^n f(X_j) \begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix} w_{ij}^{(k)}(U_k),$$

will be useful. In fact, $\overline{\nabla f}_k(X_i)$ defined as above would be the expectation of $\widehat{\nabla f}_k(X_i)$ if U_k were deterministic.

Proposition 9 *Let assumptions (A1)-(A4) be fulfilled. If for some integer $k \in [2, k(n)]$ the real number $\alpha_k = 2\delta_{k-1}^2\rho_k^{-2} + 2\delta_{k-1}\rho_k^{-1}$ is less than the constant α appearing in assumption (A3), then there exist Gaussian vectors $\xi_{1,k}^*, \dots, \xi_{L,k}^* \in \mathbb{R}^d$ such that $\max_{1\le\ell\le L}\mathbf{E}[|\xi_{\ell,k}^*|^2] \le c_0^2\sigma^2$ and*

$$\mathbf{P}\Big(\max_{1\le\ell\le L}\Big|P_k^*(\hat{\beta}_{\ell,k}-\beta_\ell) - \frac{\xi_{\ell,k}^*}{\sqrt{n}h_k}\Big| \ge \Upsilon_k,\ \hat{A}_{k-1}\in\mathscr{P}_{k-1}\Big) \le \frac{2}{n}\ ,$$

where we used the notation $\Upsilon_k = \sqrt{C_V}C_g(\rho_k+\delta_{k-1})^2 h_k + c_1\sigma\alpha_k t_n/(\sqrt{n}h_k)$ with $t_n = 4+(3\log(Ln)+\frac{3}{2}d^2\log n)^{1/2}$, $c_0 = \bar{\psi}(dC_KC_V)^{1/2}$ and $c_1 = 15\bar{\psi}(C_w^2C_V^4C_K^2 + C_V^2C_{K'}^2)^{1/2}$.

Proof Let us start with evaluating the "bias" term $|P_k^*(\bar{\beta}_{\ell,k}-\beta_\ell)|$, where the vectors $\bar{\beta}_{\ell,k}$ are defined as $\frac{1}{n}\sum_{i=1}^n \overline{\nabla f}_k(X_i)\psi_{\ell,i}$. According to the Cauchy-Schwarz inequality, it holds

$$\begin{aligned}\big|P_k^*\big(\bar{\beta}_{\ell,k}-\beta_\ell\big)\big|^2 &= n^{-2}\Big|\sum_{i=1}^n P_k^*\big(\overline{\nabla f}_k(X_i) - \nabla f(X_i)\big)\psi_{\ell,i}\Big|^2\\ &\le n^{-2}\sum_{i=1}^n\big|P_k^*\big(\overline{\nabla f}_k(X_i)-\nabla f(X_i)\big)\big|^2\sum_{i=1}^n\psi_{l,i}^2\\ &\le \max_{i=1,\dots,n}\big|P_k^*\big(\overline{\nabla f}_k(X_i)-\nabla f(X_i)\big)\big|^2.\end{aligned}$$

Simple computations show that

$$\begin{aligned}\left|P_k^*\left(\overline{\nabla f}_k(X_i)-\nabla f(X_i)\right)\right| &\le \left|\begin{pmatrix} h_k^{-1}\overline{f}_k(X_i)\\ P_k^*\overline{\nabla f}_k(X_i)\end{pmatrix}-\begin{pmatrix} h_k^{-1}f(X_i)\\ P_k^*\nabla f(X_i)\end{pmatrix}\right| \\ &= \left|h_k^{-1}V_i^{(k)}(U_k)^{-1}\sum_{j=1}^n f(X_j)\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}w_{ij}^{(k)}(U_k)-\begin{pmatrix} h_k^{-1}f(X_i)\\ P_k^*\nabla f(X_i)\end{pmatrix}\right| \\ &= h_k^{-1}\left|V_i^{(k)}(U_k)^{-1}\sum_{j=1}^n r_{ij}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}w_{ij}^{(k)}(U_k)\right| := b(X_i),\end{aligned}$$

where $r_{ij}=f(X_j)-f(X_i)-X_{ij}^\top\nabla f(X_i)$. Define $v_j=V_i^{(k)}(U_k)^{-1/2}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}\sqrt{w_{ij}^{(k)}(U_k)}$, $\lambda_j=h_k^{-1}r_{ij}\sqrt{w_{ij}^{(k)}(U_k)}$ and $\lambda=(\lambda_1,\dots,\lambda_n)^\top$. Then $\sum_j v_j v_j^\top=I_{d+1}$ and

$$b(X_i)=\left|V_i^{(k)}(U_k)^{-1/2}\sum_{j=1}^n\lambda_j v_j\right|\le\left\|V_i^{(k)}(U_k)^{-1/2}\right\|\cdot|\lambda|.$$

Note now that in view of Lemma 21, $\|U_k-I\|_2\le\alpha_k$ on the event $\{\hat{A}_{k-1}\in\mathscr{P}_{k-1}\}$. Therefore,

$$\begin{aligned}b(X_i)^2 &\le h_k^{-2}\left\|V_i^{(k)}(U_k)^{-1/2}\right\|^2\cdot\sum_{j=1}^n r_{ij}^2 w_{ij}^{(k)}(U_k)\\ &\le h_k^{-2}\max_{j:w_{ij}^{(k)}(U_k)\ne0} r_{ij}^2\left\|V_i^{(k)}(U_k)^{-1}\right\|\cdot\sum_{j=1}^n w_{ij}^{(k)}(U_k)\le C_V h_k^{-2}\max_{j:w_{ij}^{(k)}(U_k)\ne0} r_{ij}^2.\end{aligned}$$

Let us denote by Θ the $(d\times m^*)$ matrix having ϑ_l as lth column. Then $\Pi^*=\Theta\Theta^\top$ and therefore, in view of (A1),

$$\begin{aligned}|r_{ij}| &= |f(X_j)-f(X_i)-X_{ij}^\top\nabla f(X_i)|\\ &= |g(\Theta^\top X_j)-g(\Theta^\top X_i)-(\Theta^\top X_{ij})^\top\nabla g(\Theta^\top X_i)|\\ &\le C_g|\Theta^\top X_{ij}|^2=C_g|\Pi^*X_{ij}|^2.\end{aligned}$$

Since the weights $w_{ij}^{(k)}$ are defined via the kernel function K vanishing on the interval $[1,\infty[$, we have $\max_{j:w_{ij}^{(k)}(U_k)\ne0} r_{ij}^2=\max\{r_{ij}^2:|S_kX_{ij}|\le h_k\}$. By Corollary 19, the inequality $|S_kX_{ij}|\le h_k$ implies $|\Pi^*X_{ij}|\le(\rho_k+\delta_{k-1})h_k$. On the other hand, $|\Pi^*X_{ij}|\le|X_{ij}|\le|S_kX_{ij}|\le h_k$. These estimates yield $|b(X_i)|\le\sqrt{C_V}C_g\{(\rho_k+\delta_{k-1})\wedge1\}^2h_k$, and consequently,

$$\max_{\ell=1,\dots,L}\left|P_k^*\left(\overline{\beta}_{\ell,k}-\beta_\ell\right)\right|\le\max_i b(X_i)\le\sqrt{C_V}C_g\{(\rho_k+\delta_{k-1})\wedge1\}^2h_k. \tag{9}$$

Let us evaluate now the "stochastic" error $P_k^*\left(\hat{\beta}_{\ell,k}-\overline{\beta}_{\ell,k}\right)$. Define E_1 as the $d\times(d+1)$ matrix $(0\ I)$, where 0 stands for the vector all coordinates of which are zero and I is the $d\times d$ identity matrix. Using this notation, we have $P_k^*\left(\hat{\beta}_{\ell,k}-\overline{\beta}_{\ell,k}\right)=\sum_{j=1}^n c_{j,\ell}(U_k)\,\varepsilon_j$, where

$$c_{j,\ell}(U_k)=\frac{1}{nh_k}\sum_{i=1}^n E_1V_i^{(k)}(U_k)^{-1}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}w_{ij}^{(k)}(U_k)\psi_{\ell,i}.$$

Let us define $\xi^*_{\ell,k} = \sqrt{n}h_k \sum_{j=1}^n c_{j,\ell}(I)\,\varepsilon_j$. Clearly, the vectors $\xi^*_{\ell,k}$ are centered Gaussian and, in view of Lemma 22, they satisfy $\mathbf{E}[|\xi^*_{\ell,k}|^2] \le nh_k^2\sigma^2 \sum_j |c_{j,\ell}(I)|^2 \le c_0^2\sigma^2$.

By virtue of Lemma 21, on the event $\{\hat{A}_{k-1} \in \mathscr{P}_{k-1}\}$, for any $\ell = 1,\dots,L$ we have

$$\left| P_k^*(\hat{\beta}_{\ell,k} - \bar{\beta}_{\ell,k}) - \frac{\xi^*_{\ell,k}}{\sqrt{n}h_k} \right| \le \sup_{\|U-I\|_2 \le \alpha_k} \left| \sum_{j=1}^n \big(c_{j,\ell}(U) - c_{j,\ell}(I)\big)\,\varepsilon_j \right|.$$

Set $a_{j,\ell}(U) = c_{j,\ell}(U) - c_{j,\ell}(I)$. Lemma 23 and inequality (12) imply that Proposition 14 can be applied with $\kappa_0 = \frac{c_1\alpha_k}{\sqrt{n}h_k}$ and $\kappa_1 = \frac{c_1}{\sqrt{n}h_k}$. Setting $\varepsilon = 2\alpha_k/\sqrt{n}$ we get that the probability of the event

$$\left\{ \sup_{U,\ell} \left| \sum_{j=1}^n \big(c_{j,\ell}(U) - c_{j,\ell}(I)\big)\,\varepsilon_j \right| \ge \frac{c_1\sigma\alpha_k(4 + \sqrt{3\log(Ln) + 3d^2\log(\sqrt{n})})}{\sqrt{n}h_k} \right\}$$

is less than $2/n$. This completes the proof of the proposition. ■

Corollary 10 *If $nL \ge 6$ and the assumptions of Proposition 9 are fulfilled, then*

$$\mathbf{P}\left(\max_\ell \left|P_k^*(\hat{\beta}_{\ell,k} - \beta_\ell)\right| \ge \Upsilon_k + \frac{\sigma c_0 z}{\sqrt{n}h_k}, \hat{A}_{k-1} \in \mathscr{P}_{k-1} \right) \le Lze^{-\frac{z^2-1}{2}}.$$

In particular, if $nL \ge 6$, the probability of the event

$$\left\{ \max_\ell \left|P_k^*(\hat{\beta}_{\ell,k} - \beta_\ell)\right| \ge \Upsilon_k + \frac{2\sigma c_0\sqrt{\log(Ln)}}{\sqrt{n}h_k} \right\} \cap \{\hat{A}_{k-1} \in \mathscr{P}_{k-1}\}$$

does not exceed $3/n$, where Υ_k and c_0 are defined in Proposition 9.

Proof In view of Lemma 7 in Hristache et al. (2001b), we have

$$\mathbf{P}\big(\max_{\ell=1,\dots,L} \left|\xi^*_{\ell,k}\right| \ge zc_0\sigma\big) \le \sum_{\ell=1}^L \mathbf{P}\big(\left|\xi^*_{\ell,k}\right| \ge zc_0\sigma\big) \le Lze^{-(z^2-1)/2}.$$

The choice $z = \sqrt{4\log(nL)}$ leads to the desired inequality provided that $nL \ge 6$. ■

5.2 The Accuracy of the First-step Estimator

Since at the first step no information about the EDR subspace is available, we use the same bandwidth in all directions, that is the local neighborhoods are balls (and not ellipsoids) of radius h. Therefore the first step estimator $\hat{\beta}_{\ell,1}$ of the vector β_ℓ is the same as the one used in Hristache et al. (2001a).

Proposition 11 *Under assumptions (A1), (A3), (A4) and (8), for every $\ell \le L$, there exists a d-dimensional zero mean Gaussian vector $\xi^*_{\ell,1}$ so that*

$$\left| \hat{\beta}_{\ell,1} - \beta_\ell - \frac{\xi^*_{\ell,1}}{\sqrt{n}h_1} \right| \le h_1 C_g \sqrt{C_V},$$

and $\mathbf{E}|\xi^*_{\ell,1}|^2 \le d\sigma^2 C_V C_K \overline{\psi}^2$.

Proof Since P_1^* coincides by definition with the identity matrix, the arguments used in the proof of Proposition 9 apply with $S_1 = I$ and therefore $\delta_0 = \alpha_1 = 0$. More precisely, in view of (9) and $\rho_1 = 1$, we have $|\beta_{\ell,1} - \beta_\ell| \le h_1\sqrt{C_V}C_g$ for all ℓ, while in view of the relation $U_1 = I$, we have $\hat\beta_{\ell,1} - \bar\beta_{\ell,1} = \frac{1}{\sqrt{n}h_1}\xi^*_{\ell,1}$. This yields the desired result. ■

Corollary 12 *If $nL \ge 6$ and the assertions of Proposition 11 hold, then*

$$\mathbf{P}\left(\max_\ell |\hat\beta_{\ell,1} - \beta_\ell| \ge h_1 C_g\sqrt{C_V} + \frac{2\sqrt{dC_VC_K\log(nL)}\,\sigma\bar\psi}{h_1\sqrt{n}} \right) \le \frac{1}{n}.$$

Remark 13 *In order that the kernel estimator of $\nabla f(x)$ be consistent, the ball centered at x with radius h_1 should contain at least d points from $\{X_i, i = 1,\dots,n\}$. If the design is regular, this means that h_1 is at least of order $n^{-1/d}$. The optimization of the risk of $\hat\beta_{1,\ell}$ with respect to h_1 verifying $h_1 \ge n^{-1/d}$ leads to $h_1 = Const.n^{-1/(4\vee d)}$. This motivates the choice of h_1 presented in Section 3.*

5.3 Proof of Theorem 4

Recall that at the first step we use the following values of parameters: $\hat A_0 = 0$, $\rho_1 = 1$ and $h_1 = n^{-1/(d\vee 4)}$. Let us denote

$$\gamma_1 = h_1C_g\sqrt{C_V} + \frac{2\sigma\bar\psi\sqrt{2dC_VC_K\log(nL)}}{h_1\sqrt{n}}, \quad \delta_1 = 2\gamma_1\sqrt{\mu^*},$$

and introduce the event $\Omega_1 = \{\max_\ell |\hat\beta_{1,\ell} - \beta_\ell| \le \gamma_1\}$. According to Corollary 12 the probability of the event Ω_1 is at least $1 - n^{-1}$. In conjunction with Proposition 17, this implies that $\mathbf{P}(\mathrm{tr}(I - \hat A_1)\Pi^* \le \delta_1^2) \ge 1 - n^{-1}$.

Recall that for any integer $k \in [2, k(n)]$—where $k(n)$ is the total number of iterations—we use the notation $\rho_k = a_\rho\rho_{k-1}$, $h_k = a_h h_{k-1}$ and $\alpha_k = 2\delta_{k-1}^2\rho_k^{-2} + 2\delta_{k-1}\rho_k^{-1}$. Let us introduce the additional notation

$$\begin{aligned}
\gamma_k &= \frac{1}{\sqrt{n}h_k}\begin{cases}\sqrt{n}h_k\Upsilon_k + 2\sigma c_0\sqrt{\log(nL)}, & k < k(n),\\ \sqrt{n}h_k\Upsilon_k + \sigma c_0 z, & k = k(n),\end{cases}\\
\zeta_k &= 2\mu^*(\gamma_k^2\rho_k^{-2} + \sqrt{2}\,\gamma_k\rho_k^{-1}C_g),\\
\delta_k &= 2\gamma_k\sqrt{\mu^*}/\sqrt{1-\zeta_k},\\
\Omega_k &= \{\max_\ell |P_k^*(\hat\beta_{\ell,k} - \beta_\ell)| \le \gamma_k\}.
\end{aligned}$$

Combining Lemmas 24 and 25, we obtain $\mathbf{P}(\mathrm{tr}(I - \hat A_{k-1})\Pi^* > \delta_{k-1}^2) \le \mathbf{P}(\Omega_{k-1}^c)$ and therefore, using Corollary 10, we get

$$\begin{aligned}
\mathbf{P}\big(\Omega_k^c\big) &\le \mathbf{P}\Big(\max_\ell |P_k^*(\hat\beta_{\ell,k} - \beta_\ell)| > \gamma_k,\, \hat A_{k-1} \in \mathscr{P}_{k-1}\Big) + \mathbf{P}\big(\Omega_{k-1}^c\big)\\
&\le \frac{3}{n} + \mathbf{P}\big(\Omega_{k-1}^c\big), \qquad \forall\, k \le k(n) - 1.
\end{aligned}$$

Since $\mathbf{P}(\Omega_1^c) \le 1/n$, it holds $\mathbf{P}(\Omega_{k(n)-1}^c) \le (3k(n)-5)/n$ and, by virtue of Corollary 10, $\mathbf{P}(\Omega_{k(n)}^c) \le Lze^{-(z^2-1)/2} + \frac{3k(n)-5}{n}$. In conjunction with Lemma 25, this yields

$$\mathbf{P}\big(\operatorname{tr}(I-\hat{A}_{k(n)})\Pi^* > \delta_{k(n)}^2\big) \le Lze^{-(z^2-1)/2} + \frac{3k(n)-5}{n}\,. \tag{10}$$

According to Lemma 24, we have $\delta_{k(n)-2} \le \rho_{k(n)-1}$, $\alpha_{k(n)-1} \le 4$ and $\zeta_{k(n)-1} \le 1/2$. Consequently, for n sufficiently large, we have

$$\delta_{k(n)-1} = \frac{2\sqrt{\mu^*}\gamma_{k(n)-1}}{\sqrt{1-\zeta_{k(n)-1}}} \le C\left(\frac{\log(Ln)}{n}\right)^{1/2} \vee n^{-2/3\vee m^*}$$

and $\alpha_{k(n)} \le 4\delta_{k(n)-1}\rho_{k(n)}^{-1} \le C[(\sqrt{\log(Ln)}(\rho_{k(n)}\sqrt{n})^{-1}) \vee n^{-1/3\vee m^*}]$. Since $h_{k(n)} = 1$ and $(n\rho_{k(n)})^{-1} \le \rho_{k(n)}^2 = n^{-2/(3\vee m^*)}$, we infer that

$$\begin{aligned}\gamma_{k(n)} &= C_g\sqrt{C_V}\,(\rho_{k(n)} + \delta_{k(n)-1})^2 + \frac{\sigma(zc_0 + c_1\alpha_{k(n)}t_n)}{\sqrt{n}} \\ &\le Ct_n^2 n^{-2/(3\vee m^*)} + \frac{c_0\sigma z}{\sqrt{n}}.\end{aligned}$$

Therefore $\zeta_n := \zeta_{k(n)} = O(\gamma_{k(n)}\rho_{k(n)}^{-1})$ tends to zero as n tends to infinity not slower than $\sqrt{\log(nL)}\,n^{-1/(6\vee m^*)}$ and the assertion of the theorem follows from (10), the definition of $\delta_{k(n)}$ and Lemma 20.

5.4 Maximal Inequality

The following result contains a well known maximal inequality for the maximum of a Gaussian process. We include its proof for the completeness of exposition. Let $\mathbb{S}_{d-1}$ denote the unit ball of $\mathbb{R}^d$.

Proposition 14 *Let r be a positive number and let Γ be a finite set. Let functions $a_{j,\gamma} : \mathbb{R}^p \to \mathbb{R}^d$ obey the conditions*

$$\begin{aligned}&\sup_{\gamma\in\Gamma}\sup_{|u-u^*|\le r}\sum_{j=1}^n |a_{j,\gamma}(u)|^2 \le \kappa_0^2, \\ &\sup_{\gamma\in\Gamma}\sup_{|u-u^*|\le r}\sup_{e\in\mathbb{S}_{d-1}}\sum_{j=1}^n \left|\frac{d}{du}\,(e^\top a_{j,\gamma}(u))\right|^2 \le \kappa_1^2\end{aligned}$$

for some $u^* \in \mathbb{R}^p$. If the ε_j's are independent $\mathcal{N}(0,\sigma^2)$-distributed random variables, then

$$\mathbf{P}\left(\sup_{\gamma\in\Gamma}\sup_{|u-u^*|\le r}\left|\sum_{j=1}^n a_{j,\gamma}(u)\,\varepsilon_j\right| > t\sigma\kappa_0 + 2\sqrt{n}\sigma\kappa_1\varepsilon\right) \le \frac{2}{n},$$

where $t = \sqrt{3\log(|\Gamma|(2r/\varepsilon)^p n)}$ and $|\Gamma|$ is the cardinality of Γ.

Proof Let B_r be the ball $\{u : |u-u^*| \le r\} \subset \mathbb{R}^p$ and $\Sigma_{r,\varepsilon}$ be an ε-net on B_r such that for any $u \in B_r$ there is an element $u_l \in \Sigma_{r,\varepsilon}$ such that $|u-u_l| \le \varepsilon$. It is easy to see that such a net with cardinality $N_{r,\varepsilon} < (2r/\varepsilon)^p$ can be constructed. For every $u \in B_r$ we denote$\eta_\gamma(u) = \sum_{j=1}^n a_{j,\gamma}(u)\varepsilon_j$. Since $\mathbf{E}(|\eta_\gamma(u)|^2) \le \sigma^2\kappa_0^2$ for any γ and for any u, we have

$$\mathbf{P}\big(|\eta_\gamma(u_l)| > t\sigma\kappa_0\big) \le \mathbf{P}\Big(|\eta_\gamma(u_l)| > t\sqrt{\mathbf{E}(|\eta_\gamma(u_l)|^2)}\Big) \le te^{-(t^2-1)/2}.$$

Thus we get

$$\mathbf{P}\Big(\sup_{\gamma\in\Gamma}\sup_{u_l\in\Sigma_{r,\varepsilon}}\big|\eta_\gamma(u_l)\big| > t\sigma\kappa_0\Big) \le \sum_{\gamma\in\Gamma}\sum_{l=1}^{N_{r,\varepsilon}}\mathbf{P}\Big(\big|\eta_\gamma(u_l)\big| > t\sigma\kappa_0\Big) \le |\Gamma|N_{r,\varepsilon}te^{-(t^2-1)/2}.$$

Hence, if $t = \sqrt{3\log(|\Gamma|N_{r,\varepsilon}n)}$, then $\mathbf{P}\Big(\sup_{\gamma\in\Gamma}\sup_{u_l\in\Sigma_{r,\varepsilon}}\big|\eta_\gamma(u_l)\big| > t\sigma\kappa_0\Big) \le 1/n$. On the other hand, for any $u, u' \in B_r$,

$$\begin{aligned}
\big|\eta_\gamma(u)-\eta_\gamma(u')\big|^2 &= \sup_{e\in\mathbb{S}_{d-1}}\big|e^\top\big(\eta_\gamma(u)-\eta_\gamma(u')\big)\big|^2 \\
&\le |u-u'|^2\cdot\sup_{u\in B_r}\sup_{e\in\mathbb{S}_{d-1}}\left|\frac{d(e^\top\eta_\gamma)}{du}(u)\right|^2 \\
&= |u-u'|^2\cdot\sup_{u\in B_r}\sup_{e\in\mathbb{S}_{d-1}}\left|\sum_{j=1}^n\frac{d(e^\top a_{j,\gamma})}{du}(u)\varepsilon_j\right|^2.
\end{aligned}$$

The Cauchy-Schwarz inequality yields

$$\frac{\big|\eta_\gamma(u)-\eta_\gamma(u')\big|^2}{|u-u'|^2} \le \sup_{u\in B_r}\sup_{e\in\mathbb{S}_{d-1}}\sum_{j=1}^n\left|\frac{d(e^\top a_{j,\gamma})}{du}(u)\right|^2\sum_{j=1}^n\varepsilon_j^2 \le \kappa_1^2\sum_{j=1}^n\varepsilon_j^2.$$

Since $\mathbf{P}\big(\sum_{j=1}^n\varepsilon_j^2 > 4n\sigma^2\big)$ is certainly less than n^{-1}, we have

$$\begin{aligned}
&\mathbf{P}\Big(\sup_{\gamma\in\Gamma}\sup_{u\in B_r}\big|\eta_\gamma(u)\big| > t\sigma\kappa_0 + 2\sqrt{n}\sigma\kappa_1\varepsilon\Big) \\
&\quad\le \mathbf{P}\Big(\sup_{\gamma\in\Gamma}\sup_{u_l\in\Sigma_{r,\varepsilon}}\frac{|\eta_\gamma(u_l)|}{t\sigma\kappa_0} > 1\Big) + \mathbf{P}\Big(\sup_{\gamma\in\Gamma}\sup_{u\in B_r}\frac{|\eta_\gamma(u)-\eta_\gamma(u_l(u))|}{2\sqrt{n}\sigma\kappa_1\varepsilon} > 1\Big) \\
&\quad\le \frac{1}{n} + \mathbf{P}\Big(\sup_{u\in B_r}\kappa_1^2|u-u_l(u)|^2\sum_{j=1}^n\varepsilon_j^2 > 4n\sigma^2\kappa_1^2\varepsilon^2\Big) \le \frac{2}{n},
\end{aligned}$$

and the assertion of proposition follows. ■

5.5 Properties of the Solution to (6)

We collect below some simple facts concerning the solution to the optimization problem (6). By classical arguments, it is always possible to choose a measurable solution $\hat{A}$ to (6). This measurability will be assumed in the sequel.

In Proposition 15, the case of general m (not necessarily equal to m^*) is considered. As we explain below, this generality is useful for further developments of the method extending it to the case of unknown structural dimension m^*.

The vectors β_ℓ are assumed to belong to a m^*-dimensional subspace $\mathcal{S}$ of $\mathbb{R}^d$, but in this subsection we do not necessarily assume that β_ℓs are defined by (4). In fact, we will apply the results of this subsection to the vectors $\Pi^*\hat\beta_\ell$.

For every $A \in \mathcal{A}_{m^*}$, let us define

$$R(A) = \max_{1\le\ell\le L} \hat\beta_\ell^\top (I-A)\hat\beta_\ell, \quad \hat A_m = \operatorname*{arg\,min}_{A\in\mathcal{A}_m} R(A),$$
$$\hat{\mathcal{R}}(m) = \min_{A\in\mathcal{A}_m} \sqrt{R(A)} = \sqrt{R(\hat A_m)} = \min_{A\in\mathcal{A}_m} \max_{1\le\ell\le L} |(I-A)^{1/2}\hat\beta_\ell|.$$

We also define

$$\mathcal{R}^*(m) = \min_{A\in\mathcal{A}_m} \max_{1\le\ell\le L} |(I-A)^{1/2}\beta_\ell|$$

and denote by A_m^* a minimizer of $\max_\ell \beta_\ell^\top (I-A)\beta_\ell$ over $A \in \mathcal{A}_m$. Note also that for every $m \ge m^*$ the projector Π^* belongs to $\mathcal{A}_m$. Therefore, we have $A_m^* = \Pi^*$ and $\mathcal{R}^*(m) = 0$ for every $m \ge m^*$.

Proposition 15 *Let $\mathcal{B}^* = \{\bar\beta = \sum_\ell c_\ell\beta_\ell : \sum_\ell |c_\ell| \le 1\}$ be the convex hull of vectors $\pm\beta_\ell$. If $\max_\ell |\hat\beta_\ell - \beta_\ell| \le \varepsilon$, then*

$$\hat{\mathcal{R}}(m) \le \mathcal{R}^*(m) + \varepsilon,$$
$$\max_{\bar\beta\in\mathcal{B}^*} |(I-\hat A_m)^{1/2}\bar\beta| \le \mathcal{R}^*(m) + 2\varepsilon.$$

When $m < m^*$, we have also the lower bound $\hat{\mathcal{R}}(m) \ge (\mathcal{R}^*(m) - \varepsilon)_+$.

Proof For every $\ell \in 1,\dots,L$, we have

$$\begin{aligned}|(I-A_m^*)^{1/2}\hat\beta_\ell| &\le |(I-A_m^*)^{1/2}\beta_\ell| + |(I-A_m^*)^{1/2}(\hat\beta_\ell - \beta_\ell)| \\ &\le \mathcal{R}^*(m) + |\hat\beta_\ell - \beta_\ell| \le \mathcal{R}^*(m) + \varepsilon.\end{aligned}$$

Since $\hat A_m$ minimizes $\max_\ell |(I-A)^{1/2}\hat\beta_\ell|$ over $A \in \mathcal{A}_m$, we have

$$\max_\ell |(I-\hat A_m)^{1/2}\hat\beta_\ell| \le \max_\ell |(I-A_m^*)^{1/2}\hat\beta_\ell| \le \mathcal{R}^*(m) + \varepsilon.$$

Since $\hat A_m \in \mathcal{A}_m$, we have $0 \preceq (I-\hat A_m)^{1/2} \preceq I$ and consequently, for every ℓ,

$$\begin{aligned}|(I-\hat A_m)^{1/2}\beta_\ell| &\le |(I-\hat A_m)^{1/2}\hat\beta_\ell| + |(I-\hat A_m)^{1/2}(\beta_\ell - \hat\beta_\ell)| \\ &\le |(I-\hat A_m)^{1/2}\hat\beta_\ell| + |\beta_\ell - \hat\beta_\ell| \le \mathcal{R}^*(m) + 2\varepsilon.\end{aligned}$$

The second inequality of the proposition follows now from $|(I-\hat A_m)^{1/2}\bar\beta| \le \max_\ell |(I-\hat A_m)^{1/2}\beta_\ell|$ for every $\bar\beta \in \mathcal{B}^*$.

Let us prove the last assertion of the proposition. According to the definition of $\mathcal{R}^*(m)$, for every matrix $A \in \mathcal{A}_m$ there exists an index $\ell(A)$ such that $|(I-A)^{1/2}\beta_{\ell(A)}| \ge \mathcal{R}^*(m)$. In particular, $|(I-\hat A_m)^{1/2}\beta_{\ell(\hat A_m)}| \ge \mathcal{R}^*(m)$ and hence $|(I-\hat A_m)^{1/2}\hat\beta_{\ell(\hat A_m)}| \ge |(I-\hat A_m)^{1/2}\beta_{\ell(\hat A_m)}| - |\hat\beta_{\ell(\hat A_m)} - \beta_{\ell(\hat A_m)}| \ge \mathcal{R}^*(m) - \varepsilon$. ■

Remark 16 *Proposition 15 can be used for estimating the structural dimension m. Indeed,* $\hat{\mathcal{R}}(m) \le \varepsilon$ *for* $m \ge m^*$ *and* $\hat{\mathcal{R}}(m) \ge (\mathcal{R}^*(m) - \varepsilon)_+$ *for* $m < m^*$. *Therefore, it is natural to search for the smallest value* $\widehat{m}$ *of m such that the function* $\hat{\mathcal{R}}(m)$ *does not significantly decrease for* $m \ge \widehat{m}$. *The rigorous application of this heuristic argument is currently under investigation.*

From now on, we assume that the structural dimension m^* is known and we use the shortened notation $\hat{A}$ instead of $\hat{A}_{m^*}$.

Proposition 17 *If the vectors* β_ℓ *satisfy* ***(A2)*** *and* $\max_\ell |\hat{\beta}_\ell - \beta_\ell| \le \varepsilon$, *then* $\operatorname{tr}(I - \hat{A})\Pi^* \le 4\varepsilon^2\mu^*$ *and* $\operatorname{tr}[(\hat{A} - \Pi^*)^2] \le 8\varepsilon^2\mu^*$.

Proof In view of the relations $\operatorname{tr}\hat{A}^2 \le \operatorname{tr}\hat{A} \le m^*$ and $\operatorname{tr}(\Pi^*)^2 = \operatorname{tr}\Pi^* = m^*$, we have

$$\operatorname{tr}(\hat{A} - \Pi^*)^2 = \operatorname{tr}(\hat{A}^2 - \Pi^*) + 2\operatorname{tr}(I - \hat{A})\Pi^* \le 2|\operatorname{tr}(I - \hat{A})\Pi^*|.$$

Note also that the equality $\operatorname{tr}(I - \hat{A})\Pi^* = \operatorname{tr}(I - \hat{A})^{1/2}\Pi^*(I - \hat{A})^{1/2}$ implies that $\operatorname{tr}(I - \hat{A})\Pi^* \ge 0$. Now condition (7) and Proposition 15 imply

$$\begin{aligned}\operatorname{tr}(I - \hat{A})\Pi^* &= \operatorname{tr}(I - \hat{A})^{1/2}\Pi^*(I - \hat{A})^{1/2} \\ &\le \sum_{k=1}^{m^*} \mu_k \operatorname{tr}(I - \hat{A})^{1/2}\bar{\beta}_k\bar{\beta}_k^\top(I - \hat{A})^{1/2} \\ &\le \sum_{k=1}^{m^*} \mu_k \bar{\beta}_k^\top(I - \hat{A})\bar{\beta}_k \le (2\varepsilon)^2 \sum_{k=1}^{m^*} \mu_k\end{aligned}$$

and the assertion follows. ■

Lemma 18 *Let* $\operatorname{tr}(I - \hat{A})\Pi^* \le \delta^2$ *for some* $\delta > 0$. *Then for any* $x \in \mathbb{R}^d$

$$|\Pi^* x| \le |\hat{A}^{1/2}x| + \delta|x|.$$

Proof In view of the triangle inequality, $|\Pi^* x| \le |\Pi^*\hat{A}^{1/2}x| + |\Pi^*(I - \hat{A}^{1/2})x|$. On the other hand,

$$|\Pi^*(I - \hat{A}^{1/2})x|^2 \le \|\Pi^*(I - \hat{A}^{1/2})\|_2^2 \cdot |x|^2 \le \operatorname{tr}[\Pi^*(I - \hat{A}^{1/2})^2\Pi^*] \cdot |x|^2.$$

For every $A \in \mathcal{A}_m$, it obviously holds $(I - A^{1/2})^2 = I - 2A^{1/2} + A \preceq I - A$, and hence, $\operatorname{tr}\Pi^*(I - A^{1/2})^2\Pi^* \le \operatorname{tr}\Pi^*(I - A)\Pi^*$. Therefore,

$$\operatorname{tr}\Pi^*(I - \hat{A}^{1/2})^2\Pi^* \le \operatorname{tr}\Pi^*(I - \hat{A})\Pi^* = \operatorname{tr}(I - \hat{A})\Pi^* \le \delta^2$$

yielding $|\Pi^* x| \le |\Pi^*\hat{A}^{1/2}x| + \delta|x| \le |\hat{A}^{1/2}x| + \delta|x|$ as required. ■

Corollary 19 *If for some* $\rho \in (0,1)$ *and for some* $x \in \mathbb{R}^d$, *we have* $|(I + \rho^{-2}\hat{A})^{1/2}x| \le h$, *then* $|\Pi^* x| \le (\rho + \sqrt{\operatorname{tr}(I - \hat{A})\Pi^*})h$.

Proof The result follows from Lemma 18 and the inequalities $|x| \le |(I+\rho^{-2}\hat{A})^{1/2}x| \le h$ and $|\hat{A}^{1/2}x| \le \rho|(I+\rho^{-2}\hat{A})^{1/2}x| \le \rho h$. ■

Lemma 20 *Let* $\mathrm{tr}(I-\hat{A})\Pi^* \le \delta^2$ *for some* $\delta \in [0,1)$ *and let* $\widehat{\Pi}_{m^*}$ *be the orthogonal projection matrix in* $\mathbb{R}^d$ *onto the subspace spanned by the eigenvectors of* $\hat{A}$ *corresponding to its largest* m^* *eigenvalues. Then* $\mathrm{tr}(I-\widehat{\Pi}_{m^*})\Pi^* \le \delta^2/(1-\delta^2)$.

Proof Let $\hat{\lambda}_j$ and $\hat{\vartheta}_j$, $j=1,\dots,d$ be respectively the eigenvalues and the eigenvectors of $\hat{A}$. Assume that $\hat{\lambda}_1 \ge \hat{\lambda}_2 \ge \ldots \ge \hat{\lambda}_d$. Then $\hat{A} = \sum_{j=1}^d \hat{\lambda}_j \hat{\vartheta}_j \hat{\vartheta}_j^\top$ and $\widehat{\Pi}_{m^*} = \sum_{j=1}^{m^*} \hat{\vartheta}_j \hat{\vartheta}_j^\top$. Moreover, $\sum_{j=1}^d \hat{\vartheta}_j \hat{\vartheta}_j^\top = I$ since $\{\hat{\vartheta}_1,\dots,\hat{\vartheta}_d\}$ is an orthonormal basis of $\mathbb{R}^d$. This implies that

$$\begin{aligned}
\mathrm{tr}[\hat{A}\Pi^*] &\le \sum_{j\le m^*} \hat{\lambda}_j \,\mathrm{tr}[\hat{\vartheta}_j\hat{\vartheta}_j^\top\Pi^*] + \hat{\lambda}_{m^*} \sum_{j>m^*} \mathrm{tr}[\hat{\vartheta}_j\hat{\vartheta}_j^\top\Pi^*] \\
&= \sum_{j\le m^*} (\hat{\lambda}_j - \hat{\lambda}_{m^*})\,\mathrm{tr}[\hat{\vartheta}_j\hat{\vartheta}_j^\top\Pi^*] + \hat{\lambda}_{m^*}\,\mathrm{tr}\Big[\sum_{j=1}^d \hat{\vartheta}_j\hat{\vartheta}_j^\top\Pi^*\Big] \\
&= \sum_{j\le m^*} (\hat{\lambda}_j - \hat{\lambda}_{m^*})\,\mathrm{tr}[\hat{\vartheta}_j\hat{\vartheta}_j^\top\Pi^*] + m^*\hat{\lambda}_{m^*}.
\end{aligned}$$

Since $\mathrm{tr}[\hat{\vartheta}_j\hat{\vartheta}_j^\top\Pi^*] = |\Pi^*\hat{\vartheta}_j|^2 \le 1$, we get $\mathrm{tr}[\hat{A}\Pi^*] \le \sum_{j\le m^*}\hat{\lambda}_j$. Taking into account the relations $\sum_{j\le d}\hat{\lambda}_j \le m^*$, $\mathrm{tr}\,\Pi^* = m^*$ and $(1-\hat{\lambda}_{m^*+1})(I-\widehat{\Pi}_{m^*}) \preceq I-\hat{A}$, we get $\hat{\lambda}_{m^*+1} \le m^* - \sum_{j\le m^*}\hat{\lambda}_j \le \mathrm{tr}[(I-\hat{A})\Pi^*] \le \delta^2$ and therefore $I-\widehat{\Pi}_{m^*} \preceq (1-\delta^2)^{-1}(I-\hat{A})$. Consequently, $\mathrm{tr}[(I-\widehat{\Pi}_{m^*})\Pi^*] \le (1-\delta^2)^{-1}\mathrm{tr}[(I-\hat{A})\Pi^*] \le \delta^2/(1-\delta^2)$. ■

5.6 Technical Lemmas

This subsection contains five technical results. The first three lemmas have been used in the proof of Proposition 9, whereas the two last lemmas have been used in the proof of Theorem 4.

Lemma 21 *For every* $\rho \in (0,1]$ *and for every* $A \in \mathcal{A}_{m^*}$ *we have*

$$\|P_\rho^*(I+\rho^{-2}A)P_\rho^* - I\|_2 \le 2\delta_A^2\rho^{-2} + 2\delta_A\rho^{-1},$$

where $P_\rho^* = (I+\rho^{-2}\Pi^*)^{-1/2}$ *and* $\delta_A^2 = \mathrm{tr}[(I-A)\Pi^*]$.

Proof The inequality $P_\rho^* \preceq (I-\Pi^*) + \rho\Pi^*$ implies that

$$\begin{aligned}
\rho^2\big\|P_\rho^*(I+\rho^{-2}A)P_\rho^* - I\big\|_2 &= \big\|P_\rho^*(A-\Pi^*)P_\rho^*\big\|_2 \\
&\le \rho^2\big\|\Pi^*(A-\Pi^*)\Pi^*\big\|_2 + \big\|(I-\Pi^*)(A-\Pi^*)(I-\Pi^*)\big\|_2 \\
&\quad + 2\rho\big\|\Pi^*(A-\Pi^*)(I-\Pi^*)\big\|_2.
\end{aligned}$$

Since $\|B\|_2^2 = \mathrm{tr}\,BB^\top \le (\mathrm{tr}(BB^\top)^{1/2})^2$ for any matrix B, it holds

$$\begin{aligned}
\big\|\Pi^*(A-\Pi^*)\Pi^*\big\|_2 &= \big\|\Pi^*(I-A)\Pi^*\big\|_2 \\
&\le \mathrm{tr}\,\Pi^*(I-A)\Pi^* = \mathrm{tr}(I-A)\Pi^* = \delta_A^2.
\end{aligned}$$

By similar arguments one checks that

$$\begin{aligned}\left\|(I-\Pi^*)(A-\Pi^*)(I-\Pi^*)\right\|_2 &= \left\|(I-\Pi^*)A(I-\Pi^*)\right\|_2 \le \mathrm{tr}(I-\Pi^*)A \\ &= \mathrm{tr}A - \mathrm{tr}\Pi^* + \mathrm{tr}\Pi^*(I-A) \le \delta_A^2,\end{aligned}$$

and

$$\begin{aligned}\left\|\Pi^*(A-\Pi^*)(I-\Pi^*)\right\|_2 &\le \left\|\Pi^*(A-\Pi^*)\right\|_2 = \left\|\Pi^*(I-A)\right\|_2 \\ &\le \left\|\Pi^*(I-A)^{1/2}\right\|_2 = (\mathrm{tr}\Pi^*(I-A)\Pi^*)^{1/2} \\ &= (\mathrm{tr}(I-A)\Pi^*)^{1/2} = \delta_A.\end{aligned}$$

This leads to the inequality $\|P_\rho^*(I+\rho^{-2}A)P_\rho^* - I\|_2 \le \delta_A^2(1+\rho^{-2}) + 2\delta_A\rho^{-1}$, which, in view of the condition $\rho \le 1$, yields the assertion of the lemma. ■

Lemma 22 *If ψ_ℓs and U satisfy (8) and **(A3)**, then $\sum_{j=1}^n |c_{j,\ell}(U)|^2 \le dC_KC_V\overline{\psi}^2/(nh_k^2)$.*

Proof Simple computations yield

$$\sum_{j=1}^n \left| E_1V_i^{(k)}(U)^{-1}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}\right|^2 w_{ij}^{(k)}(U) = \mathrm{tr}(E_1V_i^{(k)}(U)^{-1}E_1) \le \frac{dC_V}{N_i^{(k)}(U)}. \tag{11}$$

Hence, we have

$$\begin{aligned}\sum_{j=1}^n |c_{j,\ell}|^2 &= \frac{1}{n^2h_k^2}\sum_{j=1}^n \left|\sum_{i=1}^n E_1V_i^{(k)}(U_k)^{-1}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix} w_{ij}^{(k)}(U)\,\psi_{\ell,i}\right|^2 \\ &\le \frac{\overline{\psi}^2}{n^2h_k^2}\sum_{j=1}^n \left(\sum_{i=1}^n \frac{w_{ij}^{(k)}(U)}{N_i^{(k)}(U)}\right)\left(\sum_{i=1}^n \left|E_1V_i^{(k)}(U)^{-1}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}\right|^2 N_i^{(k)}(U)w_{ij}^{(k)}(U)\right) \\ &\le \frac{C_K\overline{\psi}^2}{n^2h_k^2}\sum_{j=1}^n\sum_{i=1}^n \left|E_1V_i^{(k)}(U)^{-1}\begin{pmatrix}1\\ Z_{ij}^{(k)}\end{pmatrix}\right|^2 N_i^{(k)}(U)w_{ij}^{(k)}(U).\end{aligned}$$

Interchanging the order of summation and using inequality (11) we get the desired result. ■

Lemma 23 *If **(A3)** and (8) are fulfilled, then, for any $e \in \mathbb{S}_{d-1}$, we have*

$$\sup_{U:\|U-I\|_2\le 1/2}\max_{j=1,\dots,n}\left\|\frac{d}{dU}(e^\top c_{j,\ell})(U)\right\|_2^2 \le \frac{24C_w^2C_V^4C_K^2\overline{\psi}^2}{n^2h_k^2} + \frac{216C_V^2C_{K'}^2\overline{\psi}^2}{n^2h_k^2},$$

where $\frac{d}{dU}(e^\top c_{j,\ell})(U)$ is the $d\times d$ matrix with entries $\frac{\partial e^\top c_{j,\ell}(U)}{\partial U_{pq}}$.

Proof In order to ease the notation, we will remove the superscripts (k) in this proof. Thus, we will write V_i, w_{ij} and Z_{ij} instead of $V_i^{(k)}$, $w_{ij}^{(k)}$ and $Z_{ij}^{(k)}$. By definition of $c_{j,\ell}$ we have

$$\begin{aligned}\left\|\frac{d}{dU}\,(e^\top c_{j,\ell})(U)\right\|_2^2 &\le 2\left\|\frac{1}{nh_k}\sum_{i=1}^n\left[\frac{d}{dU}\,\tilde e^\top V_i^{-1}(U)\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\right]w_{ij}(U)\psi_{\ell,i}\right\|_2^2\\ &\quad+2\left\|\frac{1}{nh_k}\sum_{i=1}^n\tilde e^\top V_i^{-1}(U)\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\frac{dw_{ij}(U)}{dU}\,\psi_{\ell,i}\right\|_2^2\\ &=\Delta_1+\Delta_2,\end{aligned}$$

where $\tilde e = E_1^\top e$ satisfies $|\tilde e|\le|e|=1$. One checks that $d\,w_{ij}(U)/dU = \bar w_{ij}(U)Z_{ij}Z_{ij}^\top$, where we used the notation $\bar w_{ij}(U)=K'(Z_{ij}^\top UZ_{ij})$. On the one hand, $|\bar w_{ij}(U)|\cdot|Z_{ij}|^2=0$ if $Z_{ij}^\top UZ_{ij}>1$. On the other hand, the inequality $\|I-U\|_2\le 1/2$ implies that

$$|Z_{ij}|^2 \le Z_{ij}^\top UZ_{ij}+|Z_{ij}^\top(I-U)Z_{ij}| \le Z_{ij}^\top UZ_{ij}+|Z_{ij}|^2\|I-U\|_2 \le Z_{ij}^\top UZ_{ij}+|Z_{ij}|^2/2.$$

Therefore $|Z_{ij}|^2\le 2$ for all Z_{ij} verifying $Z_{ij}^\top UZ_{ij}\le 1$. Hence, $\|d\,w_{ij}(U)/dU\|_2=|\bar w_{ij}(U)|\cdot|Z_{ij}|^2\le 2|\bar w_{ij}(U)|$ and we get

$$\Delta_2 \le \frac{8\overline{\psi}^2}{n^2h_k^2}\left(\sum_{i=1}^n\left|V_i^{-1}(U)\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\bar w_{ij}(U)\right|\right)^2 \le \frac{24\overline{\psi}^2C_V^2C_{K'}^2}{n^2h_k^2}\,.$$

In order to estimate the term Δ_1, remark that the differentiation (with respect to U_{pq}) of the identity $V_i^{-1}(U)V_i(U)=I_{d+1}$ yields

$$\frac{\partial V_i^{-1}}{\partial U_{pq}}\,(U) = -V_i^{-1}(U)\,\frac{\partial V_i}{\partial U_{pq}}\,(U)V_i^{-1}(U).$$

Simple computations show that

$$\begin{aligned}\frac{\partial V_i}{\partial U_{pq}}\,(U) &= \sum_{j=1}^n\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}^\top\frac{\partial}{\partial U_{pq}}w_{ij}(U)\\ &= \sum_{j=1}^n\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}^\top\bar w_{ij}(U)(Z_{ij})_p(Z_{ij})_q.\end{aligned}$$

Hence, for any $a_1,a_2\in\mathbb{R}^{d+1}$,

$$\frac{da_1^\top V_i^{-1}(U)a_2}{dU} = \sum_{j=1}^n a_1^\top V_i^{-1}(U)\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}^\top V_i^{-1}(U)a_2\;\bar w_{ij}(U)Z_{ij}Z_{ij}^\top.$$

This relation, combined with the estimate $|Z_{ij}|^2\le 2$ for all i,j such that $\bar w_{ij}\ne 0$, implies the norm estimate

$$\begin{aligned}\left\|\frac{da_1^\top V_i^{-1}(U)a_2}{dU}\right\|_2 &\le 2\sum_{j=1}^n\left|a_1^\top V_i^{-1}(U)\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}\begin{pmatrix}1\\ Z_{ij}\end{pmatrix}^\top V_i^{-1}(U)a_2\;\bar w_{ij}(U)\right|\\ &\le 6|a_1|\,|a_2|\sum_{j=1}^n\left\|V_i^{-1}(U)\right\|^2|\bar w_{ij}(U)|\\ &\le 6C_wC_V^2|a_1|\,|a_2|N_i(U)^{-1}.\end{aligned}$$

This yields $\Delta_1 \le 216 C_w^2 C_V^4 C_K^2 \overline{\psi}^2/(nh_k)^2$ and the assertion of the lemma follows. ■

Note that under the assumptions of Lemma 23, for some $\tilde{U}$ satisfying $\|\tilde{U}-I\|_2 \le \|U-I\|_2$, it holds

$$\begin{aligned}
|c_{j,\ell}(U)-c_{j,\ell}(I)| &= \sup_{e\in\mathbb{S}_{d-1}} |e^\top (c_{j,\ell}(U)-c_{j,\ell}(I))| \\
&= \sup_{e\in\mathbb{S}_{d-1}} |\mathbf{vec}\Big[\frac{d\, e^\top c_{j,\ell}}{dU}(\tilde{U})\Big]^\top \mathbf{vec}(U-I)| \\
&\le \sup_{e\in\mathbb{S}_{d-1}} \Big\|\frac{d\, e^\top c_{j,\ell}}{dU}(\tilde{U})\Big\|_2 \|U-I\|_2 \\
&\le \frac{\sqrt{216}\,\overline{\psi}}{nh_k}\,(C_w^2 C_V^4 C_K^2 + C_V^2 C_{K'}^2)^{1/2}\|U-I\|_2, \qquad (12)
\end{aligned}$$

where $\mathbf{vec}(\cdot)$ is a matrix operator that stacks the matrix's columns one by one. In other terms, for every $d\times d$ matrix M, $\mathbf{vec}(M) = (m_{\bullet,1}^\top,\dots,m_{\bullet,d}^\top)^\top$ where $m_{\bullet,j}$ stands for the j^{th} column of M.

Lemma 24 *There exists an integer $n_0 \ge 0$ such that, for every $n \ge n_0$ and for all $k \in \{2,\dots,k(n)\}$, we have $\delta_{k-1} \le \rho_k$, $\alpha_k \le 4$ and $\zeta_k \le 1/2$.*

Proof In view of the relations $C_0 n^{-1/(d\vee 4)} = \rho_1 h_1$ and $\rho_{k(n)} h_{k(n)} \ge C_2 n^{-1/3}$, the sequence

$$s_n = 4\sqrt{C_V} C_g h_1 + \frac{4\sigma(c_0\sqrt{\log(Ln)}+c_1 t_n)}{\sqrt{n}\,\rho_{k(n)} h_{k(n)}}$$

tends to zero as $n\to\infty$.

We do now induction on k. Since $s_n \to 0$ as $n\to\infty$ and $\gamma_1 \le s_n$, the inequality $\delta_1 = 2\gamma_1\sqrt{\mu^*} \le 1/\sqrt{2} = \rho_1/\sqrt{2}$ is true for sufficiently large values of n. Let us prove the implication

$$\delta_{k-1} \le \rho_{k-1}/\sqrt{2} \quad\Longrightarrow\quad \begin{cases} \zeta_k \le 1/2, \\ \delta_k \le \rho_k/\sqrt{2}. \end{cases}$$

Since $1/\sqrt{2} \le e^{-1/6}$, the inequality $\delta_{k-1} \le \rho_k/\sqrt{2}$ entails that $\delta_{k-1} \le \rho_k$ and therefore $\alpha_k \le 4$. By our choice of a_h and a_ρ, we have $\rho_1 h_1 \ge \rho_k h_k \ge \rho_{k(n)} h_{k(n)}$. Therefore,

$$\begin{aligned}
\frac{\gamma_k}{\rho_k} &\le 4\sqrt{C_V} C_g \rho_k h_k + \frac{4\sigma(c_0\sqrt{\log(Ln)}+c_1 t_n)}{\sqrt{n}\,\rho_k h_k} \\
&\le 4\sqrt{C_V} C_g h_1 + \frac{4\sigma(c_0\sqrt{\log(Ln)}+c_1 t_n)}{\sqrt{n}\rho_{k(n)} h_{k(n)}} = s_n.
\end{aligned}$$

Thus, for n large enough, $\zeta_k \le 1/2$ and $\gamma_k \le \rho_k/4$. This implies that $\delta_k = 2\gamma_k(1-\zeta_k)^{-1/2} \le \rho_k/\sqrt{2}$.

By induction we infer that $\delta_{k-1} \le \rho_{k-1}/\sqrt{2} \le \rho_k$ and $\zeta_k \le 1/2$ for any $k = 2,\dots,k(n)-1$. This completes the proof of the lemma. ■

Lemma 25 *If $k > 2$ and $\zeta_{k-1} < 1$ then $\Omega_{k-1} \subset \{\mathrm{tr}(I - \hat{A}_{k-1})\Pi^* \le \delta_{k-1}^2\}$.*

Proof Let us denote by $\tilde{\beta}_\ell$ the vector $\Pi^* \hat{\beta}_{\ell,k-1}$, which clearly belongs to $\mathcal{S}$. It holds

$$|P_{k-1}^*(\hat{\beta}_{\ell,k-1} - \beta_\ell)| \le \gamma_{k-1} \quad \Longrightarrow \quad \begin{cases} |\hat{\beta}_{\ell,k-1} - \tilde{\beta}_\ell| \le \gamma_{k-1}, \\ |\tilde{\beta}_\ell - \beta_\ell| \le \sqrt{2}\gamma_{k-1}/\rho_{k-1}. \end{cases}$$

Set $B = \sum_{i=1}^{m^*} \mu_i \bar{\beta}_i \bar{\beta}_i^\top$ and $\tilde{B} = \sum_{i=1}^{m^*} \mu_i \bar{\tilde{\beta}}_i \bar{\tilde{\beta}}_i^\top$, where $\bar{\tilde{\beta}}_i = \sum_\ell c_\ell \tilde{\beta}_\ell$ if $\bar{\beta}_i = \sum_\ell c_\ell \beta_\ell$, see assumption (A2). Since $\sum_\ell |c_\ell| \le 1$, we have $|\bar{\beta}_i| \le \max_\ell |\beta_\ell| \le \|\nabla f\|_\infty$ and $|\bar{\beta}_i - \bar{\tilde{\beta}}_i| \le \max_\ell |\beta_\ell - \tilde{\beta}_\ell|$. Therefore

$$\begin{aligned} \|B - \tilde{B}\| &\le \sum_{i=1}^{m^*} \mu_i \|\bar{\beta}_i \bar{\beta}_i^\top - \bar{\tilde{\beta}}_i \bar{\tilde{\beta}}_i^\top\| \le \mu^* \max_i \|\bar{\beta}_i \bar{\beta}_i^\top - \bar{\tilde{\beta}}_i \bar{\tilde{\beta}}_i^\top\| \\ &\le \mu^* \max_i \Big(|\bar{\beta}_i - \bar{\tilde{\beta}}_i|^2 + 2|\bar{\beta}_i| \cdot |\bar{\beta}_i - \bar{\tilde{\beta}}_i| \Big) \\ &\le \mu^* \big(2\gamma_{k-1}^2 \rho_{k-1}^{-2} + 2\sqrt{2}\gamma_{k-1}\rho_{k-1}^{-1} \max_\ell |\beta_\ell| \big) = \zeta_{k-1} \end{aligned}$$

and hence, for every unit vector $v \in \mathcal{S}$, $v^\top \tilde{B} v \ge \big(v^\top B v - \big| v^\top B v - v^\top \tilde{B} v \big| \big) \ge v^\top B v - \|B - \tilde{B}\| \ge 1 - \zeta_{k-1}$. This inequality implies that $\Pi^* \preceq (1 - \zeta_{k-1})^{-1} \tilde{B}$. Thus the vectors $\tilde{\beta}_\ell$ satisfy assumption (A2) with μ^* replaced by $\mu^*/(1 - \zeta_{k-1})$. Applying Proposition 17 to these vectors we obtain the assertion of the lemma. ■

Acknowledgments

Much of this work has been carried out when the first author was visiting the Weierstrass Institute for Applied Analysis and Stochastics. The financial support from the institute and the hospitality of Professor Spokoiny are gratefully acknowledged.

The authors are grateful to the referees for their constructive comments, which have greatly improved the paper.

References

P. Bickel, C. Klaassen, Y. Ritov, and J. Wellner. *Efficient and Adaptive Estimation for Semiparametric Models*. Princeton University Press, Springer, New York, 1998.

E. Bura. Using linear smoothers to assess the structural dimension of regressions. *Statistica Sinica*, 13(1):143–162, 2003.

E. Bura and R. D. Cook. Estimating the structural dimension of regressions via parametric inverse regression. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 63(2):393–410, 2001a.

E. Bura and R. D. Cook. Extending sliced inverse regression: The weighted chi-squared test. *J. Amer. Statist. Assoc.*, 96(455):996–1003, 2001b.

K. S. Chan, M. C. Li, and H. Tong. Partially linear reduced-rank regression. *Technical report, available at* `www.stat.uiowa.edu/techrep/tr328.pdf`, 2004.

R. D. Cook. *Regression graphics. Ideas for studying regressions through graphics*. Wiley Series in Probability and Statistics: Probability and Statistics, John Wiley & Sons, Inc., New York, 1998.

R. D. Cook and B. Li. Dimension reduction for conditional mean in regression. *Ann. Statist.*, 30(2): 455–474, 2002.

R. D. Cook and B. Li. Determining the dimension of iterative hessian transformation. *Ann. Statist.*, 32(6):2501–2531, 2004.

R. D. Cook and L. Ni. Sufficient dimension reduction via inverse regression: a minimum discrepancy approach. *J. Amer. Statist. Assoc.*, 100(470):410–428, 2005.

R. D. Cook and L. Ni. Using intraslice covariances for improved estimation of the central subspace in regression. *Biometrika*, 93(1):65–74, 2006.

R. D. Cook and S. Weisberg. *Applied Regression Including Computing and Graphics*. Hoboken NJ: John Wiley, 1999.

R. D. Cook and S. Weisberg. Discussion of "sliced inverse regression for dimension reduction" by K. C. Li. *J. Amer. Statist. Assoc.*, 86(414):328–332, 1991.

M. Delecroix, M. Hristache, and V. Patilea. On semiparametric *m*-estimation in single-index regression. *J. Statist. Plann. Inference*, 136(3):730–769, 2006.

J. Fan and I. Gijbels. *Local polynomial modelling and its applications*. Monographs on Statistics and Applied Probability, 66, Chapman & Hall, London, 1996.

M. Hristache, A. Juditsky, J. Polzehl, and V. Spokoiny. Structure adaptive approach for dimension reduction. *Ann. Statist.*, 29(6):1537–1566, 2001a.

M. Hristache, A. Juditsky, and V Spokoiny. Direct estimation of the index coefficient in a single-index model. *Ann. Statist.*, 29(3):595–623, 2001b.

K.C. Li. On principal hessian directions for data visualization and dimension reduction: another application of stein's lemma. *J. Amer. Statist. Assoc.*, 87(420):1025–1039, 1992.

K.C. Li. Sliced inverse regression for dimension reduction. with discussion and a rejoinder by the author. *J. Amer. Statist. Assoc.*, 86(414):316–342, 1991.

K.C. Li and N. Duan. Regression analysis under link violation. *Ann. Statist.*, 17(3):1009–1052, 1989.

L. Li. Sparse sufficient dimension reduction. *Biometrika*, 94(3):603–613, 2007.

L. Ni, R. D. Cook, and C.-L. Tsai. A note on shrinkage sliced inverse regression. *Biometrika*, 92 (1):242–247, 2005.

A. Samarov, V. Spokoiny, and C. Vial. Component identification and estimation in nonlinear high-dimensional regression models by structural adaptation. *J. Amer. Statist. Assoc.*, 100(470):429–445, 2005.

H. Wang, L. Ni, and C.-L. Tsai. Improving dimension reduction via contour- projection. *Statistica Sinica*, 18:299–311, 2008.

Y. Xia. A constructive approach to the estimation of dimension reduction directions. *Ann. Statist.*, 35(6):2654–2690, 2007.

Y. Xia, H. Tong, W. K. Li, and L. X. Zhu. An adaptive estimation of dimension reduction space. *J. R. Stat. Soc. Ser. B Stat. Methodol.*, 64(3):363–410, 2002.

X. Yin and R. D. Cook. Direction estimation in single-index regressions. *Biometrika*, 92(2):371–384, 2005.

X. Yin and R. D. Cook. Dimension reduction via marginal high moments in regression. *Statist. Probab. Lett.*, 76(4):393–400, 2006.

Journal of Machine Learning Research 9 (2008) 1679-1709 Submitted 6/07; Revised 12/07; Published 8/08

Value Function Based Reinforcement Learning in Changing Markovian Environments

Balázs Csanád Csáji BALAZS.CSAJI@SZTAKI.HU
László Monostori* LASZLO.MONOSTORI@SZTAKI.HU
Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende utca 13–17, Budapest, H–1111, Hungary

Editor: Sridhar Mahadevan

Abstract

The paper investigates the possibility of applying value function based reinforcement learning (RL) methods in cases when the environment may change over time. First, theorems are presented which show that the optimal value function of a discounted Markov decision process (MDP) Lipschitz continuously depends on the immediate-cost function and the transition-probability function. Dependence on the discount factor is also analyzed and shown to be non-Lipschitz. Afterwards, the concept of (ε, δ)-MDPs is introduced, which is a generalization of MDPs and ε-MDPs. In this model the environment may change over time, more precisely, the transition function and the cost function may vary from time to time, but the changes must be bounded in the limit. Then, learning algorithms in changing environments are analyzed. A general relaxed convergence theorem for stochastic iterative algorithms is presented. We also demonstrate the results through three classical RL methods: asynchronous value iteration, Q-learning and temporal difference learning. Finally, some numerical experiments concerning changing environments are presented.

Keywords: Markov decision processes, reinforcement learning, changing environments, (ε, δ)-MDPs, value function bounds, stochastic iterative algorithms

1. Introduction

Stochastic control problems are often modeled by Markov decision processes (MDPs) that constitute a fundamental tool for computational learning theory. The theory of MDPs has grown extensively since Bellman introduced the discrete stochastic variant of the optimal control problem in 1957. These kinds of stochastic optimization problems have great importance in diverse fields, such as engineering, manufacturing, medicine, finance or social sciences. Several solution methods are known, for example, from the field of [neuro-]dynamic programming (NDP) or reinforcement learning (RL), which compute or approximate the optimal control policy of an MDP. These methods succeeded in solving many different problems, such as transportation and inventory control (Van Roy et al., 1996), channel allocation (Singh and Bertsekas, 1997), robotic control (Kalmár et al., 1998), production scheduling (Csáji and Monostori, 2006), logical games and problems from financial mathematics. Many applications of RL and NDP methods are also considered by the textbooks of Bertsekas and Tsitsiklis (1996), Sutton and Barto (1998) as well as Feinberg and Shwartz (2002).

∗. Also faculty in Mechanical Engineering at the Budapest University of Technology and Economics.

The dynamics of (Markovian) control problems can often be formulated as follows:

$$x_{t+1} = f(x_t, a_t, w_t), \tag{1}$$

where x_t is the state of the system at time $t \in \mathbb{N}$, a_t is a control action and w_t is some disturbance. There is also a cost function $g(x_t, a_t)$ and the aim is to find an optimal control policy that minimizes the [discounted] costs over time (the next section will contain the basic definitions). In many applications the calculation of a control policy should be fast and, additionally, environmental changes should also be taken into account. These two criteria are against each other. In most control applications during the computation of a control policy the system uses a *model* of the environment. The dynamics of (1) can be modeled with an MDP, but what happens when the model is wrong (e.g., if the transition function is incorrect) or the dynamics have changed? The changing of the dynamics can also be modeled as an MDP, however, including environmental changes as a higher level MDP very likely leads to problems which do not have any practically efficient solution methods.

The paper argues that if the model was "close" to the environment, then a "good" policy based on the model cannot be arbitrarily "wrong" from the viewpoint of the environment and, moreover, "slight" changes in the environment result only in "slight" changes in the optimal cost-to-go function. More precisely, the optimal value function of an MDP depends Lipschitz continuously on the cost function and the transition probabilities. Applying this result, the concept of (ε, δ)-MDPs is introduced, in which these functions are allowed to vary over time, as long as the cumulative changes remain bounded in the limit.

Afterwards, a general framework for analyzing stochastic iterative algorithms is presented. A novelty of our approach is that we allow the value function update operator to be time-dependent. Then, we apply that framework to deduce an approximate convergence theorem for time-dependent stochastic iterative algorithms. Later, with the help of this general theorem, we show relaxed convergence properties (more precisely, κ-approximation) for value function based reinforcement learning methods working in (ε, δ)-MDPs.

The main contributions of the paper can be summarized as follows:

1. We show that the optimal value function of a discounted MDP Lipschitz continuously depends on the immediate-cost function (Theorem 12). This result was already known for the case of transition-probability functions (Müller, 1996; Kalmár et al., 1998), however, we present an improved bound for this case, as well (Theorem 11). We also present value function bounds (Theorem 13) for the case of changes in the discount factor and demonstrate that this dependence is not Lipschitz continuous.

2. In order to study changing environments, we introduce (ε, δ)-MDPs (Definition 17) that are generalizations of MDPs and ε-MDPs (Kalmár et al., 1998; Szita et al., 2002). In this model the transition function and the cost function may change over time, provided that the accumulated changes remain bounded in the limit. We show (Lemma 18) that potential changes in the discount factor can be incorporated into the immediate-cost function, thus, we do not have to consider discount factor changes.

3. We investigate stochastic iterative algorithms where the value function operator may change over time. A relaxed convergence theorem for this kind of algorithm is presented (Theorem 20). As a corollary, we get an approximation theorem for value function based reinforcement learning methods in (ε, δ)-MDPs (Corollary 21).

4. Furthermore, we illustrate our results through three classical RL algorithms. Relaxed convergence properties in (ε,δ)-MDPs for asynchronous value iteration, Q-learning and temporal difference learning are deduced. Later, we show that our approach could also be applied to investigate approximate dynamic programming methods.

5. We also present numerical experiments which highlight some features of working in varying environments. First, two simple stochastic iterative algorithms, a "well-behaving" and a "pathological" one, are shown. Regarding learning, we illustrate the effects of environmental changes through two problems: scheduling and grid world.

2. Definitions and Preliminaries

Sequential decision making under the presence of uncertainties is often modeled by MDPs (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Feinberg and Shwartz, 2002). This section contains the basic definitions, the applied notations and some preliminaries.

Definition 1 *By a (finite, discrete-time, stationary, fully observable) Markov decision process (MDP) we mean a stochastic system characterized by a 6-tuple $\langle \mathbb{X}, \mathbb{A}, \mathcal{A}, p, g, \alpha \rangle$, where the components are as follows: $\mathbb{X}$ is a finite set of discrete states and $\mathbb{A}$ is a finite set of control actions. Mapping $\mathcal{A} : \mathbb{X} \to \mathcal{P}(\mathbb{A})$ is the availability function that renders a set of actions available to each state where $\mathcal{P}$ denotes the power set. The transition function is given by $p : \mathbb{X} \times \mathbb{A} \to \Delta(\mathbb{X})$, where $\Delta(\mathbb{X})$ is the set of all probability distributions over $\mathbb{X}$. Let $p(y|x,a)$ denote the probability of arrival at state y after executing action $a \in \mathcal{A}(x)$ in state x. The immediate-cost function is defined by $g : \mathbb{X} \times \mathbb{A} \to \mathbb{R}$, where $g(x,a)$ is the cost of taking action a in state x. Finally, $\alpha \in [0,1)$ denotes the discount rate.*

An interpretation of an MDP can be given, which viewpoint is often taken in RL, if we consider an agent that acts in an uncertain environment. The agent receives information about the state of the environment x, at each state x the agent is allowed to choose an action $a \in \mathcal{A}(x)$. After the action is selected, the environment moves to the next state according to the probability distribution $p(x,a)$ and the decision-maker collects its one-step cost, $g(x,a)$. The aim of the agent is to find an optimal behavior (policy), such that applying this strategy minimizes the expected cumulative costs.

Definition 2 *A (stationary, Markovian) control policy determines the action to take in each state. A deterministic policy, $\pi : \mathbb{X} \to \mathbb{A}$, is simply a function from states to control actions. A randomized policy, $\pi : \mathbb{X} \to \Delta(\mathbb{A})$, is a function from states to probability distributions over actions. We denote the probability of executing action a in state x by $\pi(x)(a)$ or, for short, by $\pi(x,a)$. Unless indicated otherwise, we consider randomized policies.*

For any $\widetilde{x}_0 \in \Delta(\mathbb{X})$ initial probability distribution of the states, the transition probabilities p together with a control policy π completely determine the progress of the system in a stochastic sense, namely, they define a *homogeneous Markov chain* on $\mathbb{X}$,

$$\widetilde{x}_{t+1} = P(\pi)\widetilde{x}_t,$$

where $\widetilde{x}_t$ is the state probability distribution vector of the system at time t and $P(\pi)$ denotes the probability transition matrix induced by control policy π,

$$[P(\pi)]_{x,y} = \sum_{a \in \mathbb{A}} p(y|x,a)\,\pi(x,a).$$

Definition 3 *The value or cost-to-go function of a policy π is a function from states to costs, $J^\pi : \mathbb{X} \to \mathbb{R}$. Function $J^\pi(x)$ gives the expected value of the cumulative (discounted) costs when the system is in state x and it follows policy π thereafter,*

$$J^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{N} \alpha^t g(X_t, A_t^\pi) \,\middle|\, X_0 = x\right], \tag{2}$$

where X_t and A_t^π are random variables, A_t^π is selected according to control policy π and the distribution of X_{t+1} is $p(X_t, A_t^\pi)$. The horizon of the problem is denoted by $N \in \mathbb{N} \cup \{\infty\}$. Unless indicated otherwise, we will always assume that the horizon is infinite, $N = \infty$.

Definition 4 *We say that $\pi_1 \leq \pi_2$ if and only if $\forall x \in \mathbb{X} : J^{\pi_1}(x) \leq J^{\pi_2}(x)$. A control policy is (uniformly) optimal if it is less than or equal to all other control policies.*

There always exists at least one optimal policy (Sutton and Barto, 1998). Although there may be many optimal policies, they all share the same unique optimal cost-to-go function, denoted by J^*. This function must satisfy the Bellman optimality equation (Bertsekas and Tsitsiklis, 1996), $TJ^* = J^*$, where T is the *Bellman operator*, defined for all $x \in \mathbb{X}$, as

$$(TJ)(x) = \min_{a \in \mathcal{A}(x)} \Big[g(x,a) + \alpha \sum_{y \in \mathbb{X}} p(y \,|\, x, a) J(y) \Big].$$

Definition 5 *We say that function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$, $\mathcal{Y}$ are normed spaces, is Lipschitz continuous if there exists a $\beta \geq 0$ such that $\forall x_1, x_2 \in \mathcal{X} : \|f(x_1) - f(x_2)\|_\mathcal{Y} \leq \beta \|x_1 - x_2\|_\mathcal{X}$, where $\|\cdot\|_\mathcal{X}$ and $\|\cdot\|_\mathcal{Y}$ denote the norm of $\mathcal{X}$ and $\mathcal{Y}$, respectively. The smallest such β is called the Lipschitz constant of f. Henceforth, assume that $\mathcal{X} = \mathcal{Y}$. If the Lipschitz constant $\beta < 1$, then the function is called a contraction. A mapping is called a pseudo-contraction if there exists an $x^* \in \mathcal{X}$ and a $\beta \geq 0$ such that $\forall x \in \mathcal{X}$, we have $\|f(x) - x^*\|_\mathcal{X} \leq \beta \|x - x^*\|_\mathcal{X}$.*

Naturally, every contraction mapping is also a pseudo-contraction, however, the opposite is not true. The pseudo-contraction condition implies that x^* is the fixed point of function f, namely, $f(x^*) = x^*$, moreover, x^* is unique, thus, f cannot have other fixed points.

It is known that the Bellman operator is a supremum norm contraction with Lipschitz constant α. In case we consider *stochastic shortest path* (SSP) problems, which arise if the MDP has an absorbing terminal (goal) state, then the Bellman operator becomes a pseudo-contraction in the weighted supremum norm (Bertsekas and Tsitsiklis, 1996).

From a given value function J, it is straightforward to get a policy, for example, by applying a *greedy* and deterministic policy (w.r.t. J), that always selects actions with minimal costs,

$$\pi(x) \in \operatorname*{arg\,min}_{a \in \mathcal{A}(x)} \Big[g(x,a) + \alpha \sum_{y \in \mathbb{X}} p(y \,|\, x, a) J(y) \Big].$$

Similarly to the definition of J^π, one can define *action-value* functions of control polices,

$$Q^\pi(x,a) = \mathbb{E}\left[\sum_{t=0}^{N} \alpha^t g(X_t, A_t^\pi) \,\middle|\, X_0 = x, A_0^\pi = a\right],$$

where the notations are the same as in (2). MDPs have an extensively studied theory and there exist a lot of exact and approximate solution methods, for example, value iteration, policy iteration, the Gauss-Seidel method, Q-learning, Q(λ), SARSA and TD(λ)—temporal difference learning (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Feinberg and Shwartz, 2002). Most of these reinforcement learning algorithms work by iteratively approximating the optimal value function and typically consider stationary environments.

If J is "close" to J^*, then the greedy policy with one-stage lookahead based on J will also be "close" to an optimal policy, as it was proven by Bertsekas and Tsitsiklis (1996):

Theorem 6 *Let M be a discounted MDP and J is an arbitrary value function. The value function of the greedy policy based on J is denoted by J^π. Then, we have*

$$\|J^\pi - J^*\|_\infty \leq \frac{2\alpha}{1-\alpha} \|J - J^*\|_\infty,$$

where $\|\cdot\|_\infty$ denotes the supremum norm, namely $\|f\|_\infty = \sup\{|f(x)| : x \in domain(f)\}$. Moreover, there exists an $\varepsilon > 0$ such that if $\|J - J^*\|_\infty < \varepsilon$ then $J^* = J^\pi$.

Consequently, if we could obtain a good approximation of the optimal value function, then we immediately had a good control policy, as well, for example, the greedy policy with respect to our approximate value function. Therefore, the main question for most RL approaches is that how a good approximation of the optimal value function could be achieved.

3. Asymptotic Bounds for Generalized Value Iteration

In this section we will briefly overview a unified framework to analyze value function based reinforcement learning algorithms. We will use this approach later when we prove convergence properties in changing environments. The theory presented in this section was developed by Szepesvári and Littman (1999) and was extended by Szita et al. (2002).

3.1 Generalized Value Functions and Approximate Convergence

Throughout the paper we denote the set of value functions by $\mathcal{V}$ which contains, in general, all bounded real-valued functions over an arbitrary set $\mathcal{X}$, for example, $\mathcal{X} = \mathbb{X}$, in the case of state-value functions, or $\mathcal{X} = \mathbb{X} \times \mathbb{A}$, in the case of action-value functions. Note that the set of value functions, $\mathcal{V} = \mathcal{B}(\mathcal{X})$, where $\mathcal{B}(\mathcal{X})$ denotes the set of all bounded real-valued functions over set $\mathcal{X}$, is a normed space, for example, with the supremum norm. Naturally, bounded functions constitute no real restriction in case of analyzing finite MDPs.

Definition 7 *We say that a sequence of random variables, denoted by X_t, κ-approximates random variable X with $\kappa \geq 0$, in a given norm, if we have*

$$\mathbb{P}\left(\limsup_{t\to\infty} \|X_t - X\| \leq \kappa\right) = 1. \tag{3}$$

Hence, the "meaning" of this definition is that sequence X_t converges almost surely to an environment of X and the radius of this environment is less than or equal to a given constant κ. Naturally, this definition is weaker (more general) than the probability one convergence, which arises as a special case, when $\kappa = 0$. In the paper we will always consider convergence in the supremum norm.

3.2 Relaxed Convergence of Generalized Value Iteration

A general form of value iteration type algorithms can be given as follows,

$$V_{t+1} = H_t(V_t, V_t),$$

where H_t is a random operator on $\mathcal{V} \times \mathcal{V} \to \mathcal{V}$ (Szepesvári and Littman, 1999). Consider, for example, the SARSA (state-action-reward-state-action) algorithm which is a model-free policy evaluation method. It aims at finding Q^π for a given policy π and it is defined as

$$Q_{t+1}(x,a) = (1-\gamma_t(x,a))\,Q_t(x,a) + \gamma_t(x,a)(g(x,a) + \alpha\, Q_t(Y,B)),$$

where $\gamma_t(x,a)$ denotes the stepsize associated with state x and action a at time t; Y and B are random variables, Y is generated from the pair (x,a) by simulation, that is, according to the distribution $p(x,a)$, and the distribution of B is $\pi(Y)$. In this case, H_t is defined as

$$H_t(Q_a, Q_b)(x,a) = (1-\gamma_t(x,a))\,Q_a(x,a) + \gamma_t(x,a)(g(x,a) + \alpha\, Q_b(Y,B)), \tag{4}$$

for all x and a. Therefore, the SARSA algorithm takes the form $Q_{t+1} = H_t(Q_t, Q_t)$.

Definition 8 *We say that the operator sequence H_t κ-approximates operator $H : \mathcal{V} \to \mathcal{V}$ at $V \in \mathcal{V}$ if for any initial $V_0 \in \mathcal{V}$ the sequence $V_{t+1} = H_t(V_t, V)$ κ-approximates HV.*

The next theorem (Szita et al., 2002) will be an important tool for proving convergence results for value function based RL algorithms in varying environments.

Theorem 9 *Let H be an arbitrary mapping with fixed point V^*, and let H_t κ-approximate H at V^* over set $\mathcal{X}$. Additionally, assume that there exist random functions $0 \le F_t(x) \le 1$ and $0 \le G_t(x) \le 1$ satisfying the four conditions below with probability one*

1. *For all $V_1, V_2 \in \mathcal{V}$ and for all $x \in \mathcal{X}$,*

$$|H_t(V_1, V^*)(x) - H_t(V_2, V^*)(x)| \le G_t(x)\,|V_1(x) - V_2(x)|\,.$$

2. *For all $V_1, V_2 \in \mathcal{V}$ and for all $x \in \mathcal{X}$,*

$$|H_t(V_1, V^*)(x) - H_t(V_1, V_2)(x)| \le F_t(x)\,\|V^* - V_2\|_\infty\,.$$

3. *For all $k > 0$, $\prod_{t=k}^{n} G_t(x)$ converges to zero uniformly in x as n increases.*

4. *There exist $0 \le \xi < 1$ such that for all $x \in \mathcal{X}$ and sufficiently large t,*

$$F_t(x) \le \xi\,(1 - G_t(x)).$$

Then, $V_{t+1} = H_t(V_t, V_t)$ κ'-approximates V^* over $\mathcal{X}$ for any $V_0 \in \mathcal{V}$, where $\kappa' = 2\kappa/(1-\xi)$.

Usually, functions F_t and G_t can be interpreted as the ratio of mixing the two arguments of operator H_t. In the case of the SARSA algorithm, described above by (4), $\mathcal{X} = \mathbb{X} \times \mathbb{A}$, $G_t(x,a) = (1-\gamma_t(x,a))$ and $F_t(x,a) = \alpha\gamma_t(x,a)$ would be a suitable choice.

One of the most important aspects of this theorem is that it shows how to reduce the problem of approximating V^* with $V_t = H_t(V_t, V_t)$ type operators to the problem of approximating it with a $V_t' = H_t(V_t', V^*)$ sequence, which is, in many cases, much easier to be dealt with. This makes, for example, the convergence of Watkins' Q-learning a consequence of the classical Robbins-Monro theory (Szepesvári and Littman, 1999; Szita et al., 2002).

4. Value Function Bounds for Environmental Changes

In many control problems it is typically not possible to "practise" in the real environment, only a dynamic *model* is available to the system and this model can be used for predicting how the environment will respond to the control signals (model predictive control). MDP based solutions usually work by *simulating* the environment with the model, through simulation they produce *simulated experience* and by *learning* from these experience they improve their value functions. Computing an approximately optimal value function is essential because, as we have seen (Theorem 6), close approximations to optimal value functions lead directly to good policies. Though, there are alternative approaches which directly approximate optimal control policies (see Sutton et al., 2000). However, what happens if the model was inaccurate or the environment had changed slightly? In what follows we investigate the effects of environmental changes on the optimal value function. For continuous Markov processes questions like these were already analyzed (Gordienko and Salem, 2000; Favero and Runggaldier, 2002; de Oca et al., 2003), hence, we will focus on *finite* MDPs.

The theorems of this section have some similarities with two previous results. First, Munos and Moore (2000) studied the dependence of the Bellman operator on the transition-probabilities and the immediate-costs. Later, Kearns and Singh (2002) applied a *simulation lemma* to deduce polynomial time bounds to achieve near-optimal return in MDPs. This lemma states that if two MDPs differ only in their transition and cost functions and we want to approximate the value function of a fixed policy concerning one of the MDPs in the other MDP, then how close should we choose the transitions and the costs to the original MDP relative to the mixing time or the horizon time.

4.1 Changes in the Transition-Probability Function

First, we will see that the optimal value function of a *discounted* MDP Lipschitz continuously depends on the transition-probability function. This question was analyzed by Müller (1996), as well, but the presented version of Theorem 10 was proven by Kalmár et al. (1998).

Theorem 10 *Assume that two discounted MDPs differ only in their transition functions, denoted by p_1 and p_2. Let the corresponding optimal value functions be J_1^* and J_2^*, then*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{n\alpha\|g\|_\infty}{(1-\alpha)^2}\|p_1 - p_2\|_\infty,$$

recall that n is the size of the state space and $\alpha \in [0,1)$ is the discount rate.

A disadvantage of this theorem is that the estimation heavily depends on the size of the state space, *n*. However, this bound can be improved if we consider an induced matrix norm for transition-probabilities instead of the supremum norm. The following theorem presents our improved estimation for transition changes. Its proof can be found in the appendix.

Theorem 11 *With the assumptions and notations of Theorem 10, we have*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{\alpha\|g\|_\infty}{(1-\alpha)^2}\|p_1 - p_2\|_1,$$

where $\|\cdot\|_1$ is a norm on $f : \mathbb{X}\times\mathbb{A}\times\mathbb{X} \to \mathbb{R}$ type functions, for example, $f(x,a,y) = p(y|x,a)$,

$$\|f\|_1 = \max_{x,a}\sum_{y\in\mathbb{X}} |f(x,a,y)|. \tag{5}$$

If we consider f as a matrix which has a column for each state-action pair $(x,a) \in \mathbb{X} \times \mathbb{A}$ and a row for each state $y \in \mathbb{X}$, then the above definition gives us the usual "maximum absolute column sum norm" definition for matrices, which is conventionally denoted by $\|\cdot\|_1$.

It is easy to see that for all f, we have $\|f\|_1 \leq n\|f\|_\infty$, where n is size of the state space. Therefore, the estimation of Theorem 11 is at least as good as the estimation of Theorem 10. In order to see that it is a real improvement consider, for example, the case when we choose a particular state-action pair, $(\hat{x},\hat{a})$, and take a p_1 and p_2 that only differ in $(\hat{x},\hat{a})$. For example, $p_1(\hat{x},\hat{a}) = \langle 1,0,0,\ldots,0\rangle$ and $p_2(\hat{x},\hat{a}) = \langle 0,1,0,\ldots,0\rangle$, and they are equal for all other $(x,a) \neq (\hat{x},\hat{a})$. Then, by definition, $\|p_1 - p_2\|_1 = 2$, but $n\|p_1 - p_2\|_\infty = n$. Consequently, in this case, we have improved the bound of Theorem 10 by a factor of $2/n$.

4.2 Changes in the Immediate-Cost Function

The same kind of Lipschitz continuity can be proven in case of changes in the cost function.

Theorem 12 *Assume that two discounted MDPs differ only in the immediate-costs functions, g_1 and g_2. Let the corresponding optimal value functions be J_1^* and J_2^*, then*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{1}{1-\alpha}\|g_1 - g_2\|_\infty .$$

4.3 Changes in the Discount Factor

The following theorem shows that the change of the value function can also be estimated in case there were changes in the discount rate (all proofs can be found in the appendix).

Theorem 13 *Assume that two discounted MDPs differ only in the discount factors, denoted by $\alpha_1, \alpha_2 \in [0,1)$. Let the corresponding optimal value functions be J_1^* and J_2^*, then*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{|\alpha_1 - \alpha_2|}{(1-\alpha_1)(1-\alpha_2)}\|g\|_\infty .$$

The next example demonstrates, however, that this dependence is not Lipschitz continuous. Consider, for example, an MDP that has only one state x and one action a. Taking action a loops back deterministically to state x with cost $g(x,a) = 1$. Suppose that the MDP has discount factor $\alpha_1 = 0$, thus, $J_1^*(x) = 1$. Now, if we change the discount rate to $\alpha_2 \in (0,1)$, then $|\alpha_1 - \alpha_2| < 1$ but $\|J_1^* - J_2^*\|_\infty$ could be arbitrarily large, since $J_2^*(x) \to \infty$ as $\alpha_2 \to 1$.

At the same time, we can notice that if we fix a constant $\alpha_0 < 1$ and only allow discount factors from the interval $[0,\alpha_0]$, then this dependence became Lipschitz continuous, as well.

4.4 Case of Action-Value Functions

Many reinforcement learning algorithms, such as Q-learning, work with action-value functions which are important, for example, for model-free approaches. Now, we investigate how the previously presented theorems apply to this type of value functions. The optimal action-value function, denoted by Q^*, is defined for all state-action pair (x,a) by

$$Q^*(x,a) = g(x,a) + \alpha \sum_{y \in \mathbb{X}} p(y|x,a)J^*(y),$$

where J^* is the optimal state-value function. Note that in the case of the optimal action-value function, first, we take a given action (which can have very high cost) and, only after that the action was taken, follow an optimal policy. Thus, we can estimate $\|Q^*\|_\infty$ by

$$\|Q^*\|_\infty \leq \|g\|_\infty + \alpha \|J^*\|_\infty .$$

Nevertheless, the next lemma shows that the same estimations can be derived for environmental changes in the case of action-value functions as in the case of state-value functions.

Lemma 14 *Assume that we have two discounted MDPs which differ only in the transition-probability functions or only in the immediate-cost functions or only in the discount factors. Let the corresponding optimal action-value functions be Q_1^* and Q_2^*, respectively. Then, the bounds for $\|J_1^* - J_2^*\|_\infty$ of Theorems 11, 12 and 13 are also bounds for $\|Q_1^* - Q_2^*\|_\infty$.*

4.5 Further Remarks on Inaccurate Models

In this section we saw that the optimal value function of a discounted MDP depends smoothly on the transition function, the cost function and the discount rate. This dependence is of Lipschitz type in the first two cases and non-Lipschitz for discount rates.

If we treat one of the MDPs in the previous theorems as a system which describes the "real" behavior of the environment and the other MDP as our model, then these results show that even if the model is slightly inaccurate or there were changes in the environment, the optimal value function based on the model cannot be arbitrarily wrong from the viewpoint of the environment. These theorems are of special interest because in "real world" problems the transition-probabilities and the immediate-costs are mostly estimated only, for example, by statistical methods from historical data. Later, we will see that changes in the discount rate can be traced back to changes in the cost function (Lemma 18), therefore, it is sufficient to consider transition and cost changes. The following corollary summarizes the results.

Corollary 15 *Assume that two discounted MDPs (E and M) differ only in their transition functions and their cost functions. Let the corresponding transition and cost functions be denoted by p_E, p_M and g_E, g_M, respectively. The corresponding optimal value functions are denoted by J_E^* and J_M^*. The value function in E of the deterministic and greedy policy (π) with one stage-lookahead that is based upon J_M^* is denoted by J_E^π. Then,*

$$\|J_E^\pi - J_E^*\|_\infty \leq \frac{2\alpha}{1-\alpha}\left[\frac{\|g_E - g_M\|_\infty}{1-\alpha} + \frac{c\,\alpha\,\|p_E - p_M\|_1}{(1-\alpha)^2}\right],$$

where $c = \min\{\|g_E\|_\infty, \|g_M\|_\infty\}$ and $\alpha \in [0,1)$ is the discount factor.

The proof simply follows from Theorems 6, 11 and 12 and from the triangle inequality. Another interesting question is the effects of environmental changes on the value function of a *fixed* control policy. However, it is straightforward to prove (Csáji, 2008) that the same estimations can be derived for $\|J_1^\pi - J_2^\pi\|_\infty$, where π is an arbitrary (stationary, Markovian, randomized) control policy, as the estimations of Theorems 10, 11, 12 and 13.

Note that the presented theorems are only valid in case of *discounted* MDPs. Though, a large part of the MDP related research studies the expected total discounted cost optimality criterion, in

some cases discounting is inappropriate and, therefore, there are alternative optimality approaches, as well. A popular alternative approach is to optimize the *expected average cost* (Feinberg and Shwartz, 2002). In this case the value function is defined as

$$J^\pi(x) = \limsup_{N\to\infty} \frac{1}{N}\,\mathbb{E}\left[\sum_{t=0}^{N-1} \alpha^t g(X_t, A_t^\pi) \;\middle|\; X_0 = x\right],$$

where the notations are the same as previously, for example, as applied in Equation (2).

Regarding the validity of the results of Section 4 concerning MDPs with the expected average cost minimization objective, we can recall that, in the case of finite MDPs, discounted cost offers a good approximation to the other optimality criterion. More precisely, it can be shown that there exists a large enough $\alpha_0 < 1$ such that $\forall\,\alpha \in (\alpha_0, 1)$ optimal control policies for the discounted cost problem are also optimal for the average cost problem (Feinberg and Shwartz, 2002). These policies are called *Blackwell optimal*.

5. Learning in Varying Environments

In this section we investigate how value function based learning methods can act in environments which may change over time. However, without restrictions, this approach would be too general to establish convergence results. Therefore, we restrict ourselves to the case when the changes remain bounded over time. In order to precisely define this concept, the idea of (ε, δ)-MDPs is introduced, which is a generalization of classical MDPs and ε-MDPs. First, we recall the definition of ε-MDPs (Kalmár et al., 1998; Szita et al., 2002).

Definition 16 *A sequence of MDPs $(\mathcal{M}_t)_{t=1}^\infty$ is called an ε-MDP with $\varepsilon > 0$ if the MDPs differ only in their transition-probability functions, denoted by p_t for $\mathcal{M}_t$, and there exists an MDP with transition function p, called the base MDP, such that $\sup_t \|p - p_t\| \leq \varepsilon$.*

5.1 Varying Environments: (ε, δ)-MDPs

Now, we extend the idea described above. The following definition of (ε, δ)-MDPs generalizes the concept of ε-MDPs in two ways. First, we also allow the cost function to change over time and, additionally, we require the changes to remain bounded by parameters ε and δ only asymptotically, in the limit. A finite number of large deviations is tolerated.

Definition 17 *A tuple $\langle \mathbb{X}, \mathbb{A}, \mathcal{A}, \{p_t\}_{t=1}^\infty, \{g_t\}_{t=1}^\infty, \alpha\rangle$ is an (ε, δ)-MDP with $\varepsilon, \delta \geq 0$, if there exists an MDP $\langle \mathbb{X}, \mathbb{A}, \mathcal{A}, p, g, \alpha\rangle$, called the base MDP, such that*

1. $\limsup_{t\to\infty} \|p - p_t\| \leq \varepsilon$
2. $\limsup_{t\to\infty} \|g - g_t\| \leq \delta$

The optimal value function of the base MDP and of the current MDP at time t (which MDP has transition function p_t and cost function g_t) are denoted by J^* and J_t^*, respectively.

In order to keep the analysis as simple as possible, we do not allow the discount rate parameter α to change over time; not only because, for example, with Theorem 13 at hand, it would be straightforward to extend the results to the case of changing discount factors, but even more because, as

Lemma 18 demonstrates, the effects of changes in the discount rate can be incorporated into the immediate-cost function, which is allowed to change in (ε,δ)-MDPs.

Lemma 18 *Assume that two discounted MDPs, $\mathcal{M}_1$ and $\mathcal{M}_2$, differ only in the discount factors, denoted by α_1 and α_2. Then, there exists an MDP, denoted by $\mathcal{M}_3$, such that it differs only in the immediate-cost function from $\mathcal{M}_1$, thus its discount factor is α_1, and it has the same optimal value function as $\mathcal{M}_2$. The immediate-cost function of $\mathcal{M}_3$ is*

$$\widehat{g}(x,a) = g(x,a) + (\alpha_2 - \alpha_1) \sum_{y\in\mathbb{X}} p(y|x,a) J_2^*(y),$$

where p is the probability-transition function of $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$; g is the immediate-cost function of $\mathcal{M}_1$ and $\mathcal{M}_2$; and $J_2^*(y)$ denotes the optimal cost-to-go function of $\mathcal{M}_2$.

On the other hand, we can notice that changes in the cost function cannot be traced back to changes in the transition function. Consider, for example, an MDP with a constant zero cost function. Then, no matter what the transition-probabilities are, the optimal value function remains zero. However, we may achieve non-zero optimal value function values if we change the immediate-cost function. Therefore, (ε,δ)-MDPs cannot be traced back to ε-MDPs.

Now, we briefly investigate the applicability of (ε,δ)-MDPs and a possible motivation behind them. When we model a "real world" problem as an MDP, then we typically take only the *major characteristics* of the system into account, but there could be many *hidden parameters*, as well, which may affect the transition-probabilities and the immediate-costs, however, which are not explicitly included in the model. For example, if we model a production control system as an MDP (Csáji and Monostori, 2006), then the workers' fatigue, mood or the quality of the materials may affect the durations of the tasks, but these characteristics are usually not included in the model. Additionally, the values of these hidden parameters may change over time. In these cases, we could either try to incorporate as many aspects of the system as possible into the model, which would most likely lead to *computationally intractable* results, or we could model the system as an (ε,δ)-MDP, which would result in a simplified model and, presumably, in a more tractable system.

5.2 Relaxed Convergence of Stochastic Iterative Algorithms

In this section we present a general relaxed convergence theorem for a large class of stochastic iterative algorithms. Later, we will apply this theorem to investigate the convergence properties of value function based reinforcement learning methods in (ε,δ)-MDPs.

Many learning and optimization methods can be written in a general form as a *stochastic iterative algorithm* (Bertsekas and Tsitsiklis, 1996). More precisely, as

$$V_{t+1}(x) = (1-\gamma_t(x))V_t(x) + \gamma_t(x)((K_tV_t)(x) + W_t(x)), \tag{6}$$

where $V_t \in \mathcal{V}$, operator $K_t : \mathcal{V} \to \mathcal{V}$ acts on value functions, each $\gamma_t(x)$ is a random variable which determines the stepsize and $W_t(x)$ is also a random variable, a noise parameter.

Regarding reinforcement learning algorithms, for example, (asynchronous) value iteration, Gauss-Seidel methods, Q-learning and TD(λ) – temporal difference learning can be formulated this way. We will show that under suitable conditions these algorithms work in (ε,δ)-MDPs, more precisely, κ-approximation to the optimal value function of the base MDP will be proven.

Now, in order to provide our relaxed convergence result, we introduce assumptions on the noise parameters, on the stepsize parameters and on the value function operators.

Definition 19 *We denote the history of the algorithm until time t by $\mathcal{F}_t$, defined as*

$$\mathcal{F}_t = \{V_0, \ldots, V_t, W_0, \ldots, W_{t-1}, \gamma_0, \ldots, \gamma_t\}.$$

The sequence $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots$ can be seen as a *filtration*, viz., as an increasing sequence of σ-fields. The set $\mathcal{F}_t$ represents the information available at each time t.

Assumption 1 *There exits a constant $C > 0$ such that for all state x and time t, we have*

$$\mathbb{E}[W_t(x) \,|\, \mathcal{F}_t] = 0 \quad \textit{and} \quad \mathbb{E}\left[W_t^2(x) \,|\, \mathcal{F}_t\right] < C < \infty.$$

Regarding the stepsize parameters, γ_t, we make the "usual" stochastic approximation assumptions. Note that there is a separate stepsize parameter for each possible state.

Assumption 2 *For all x and t, $0 \leq \gamma_t(x) \leq 1$, and we have with probability one*

$$\sum_{t=0}^{\infty} \gamma_t(x) = \infty \quad \textit{and} \quad \sum_{t=0}^{\infty} \gamma_t^2(x) < \infty.$$

Intuitively, the first requirement guarantees that the stepsizes are able to overcome the effects of finite noises, while the second criterion ensures that they eventually converge.

Assumption 3 *For all t, operator $K_t : \mathcal{V} \to \mathcal{V}$ is a supremum norm contraction mapping with Lipschitz constant $\beta_t < 1$ and with fixed point V_t^*. Formally, for all $V_1, V_2 \in \mathcal{V}$,*

$$\|K_t V_1 - K_t V_2\|_\infty \leq \beta_t \|V_1 - V_2\|_\infty.$$

Let us introduce a common Lipschitz constant $\beta_0 = \limsup_{t\to\infty} \beta_t$, and assume that $\beta_0 < 1$.

Because our aim is to analyze changing environments, each K_t operator can have different fixed points and different Lipschitz constants. However, to avoid the progress of the algorithm to "slow down" infinitely, we should require that $\limsup_{t\to\infty} \beta_t < 1$. In the next section, when we apply this theory to the case of (ε, δ)-MDPs, each value function operator can depend on the current MDP at time t and, thus, can have different fixed points.

Now, we present a theorem (its proof can be found in the appendix) that shows how the function sequence generated by iteration (6) can converge to an environment of a function.

Theorem 20 *Suppose that Assumptions 1-3 hold and let V_t be the sequence generated by iteration (6). Then, for any $V^*, V_0 \in \mathcal{V}$, the sequence V_t κ-approximates function V^* with*

$$\kappa = \frac{4\rho}{1-\beta_0} \quad \textit{where} \quad \rho = \limsup_{t\to\infty} \|V_t^* - V^*\|_\infty.$$

This theorem is very general, it is valid even in the case of non-finite MDPs. Notice that V^* can be an *arbitrary* function but, naturally, the radius of the environment of V^*, which the sequence V_t almost surely converges to, depends on $\limsup_{t\to\infty} \|V_t^* - V^*\|_\infty$.

If we take a closer look at the proof, we can notice that the theorem is still valid if each K_t is only a *pseudo-contraction* but, additionally, it also attracts points to V^*. Formally, it is enough if we assume that for all $V \in \mathcal{V}$, we have $\|K_t V - K_t V_t^*\|_\infty \leq \beta_t \|V - V_t^*\|_\infty$ and $\|K_t V - K_t V^*\|_\infty \leq \beta_t \|V - V^*\|_\infty$ for a suitable $\beta_t < 1$. This remark could be important in case we want to apply Theorem 20 to changing *stochastic shortest path (SSP)* problems.

5.2.1 A SIMPLE NUMERICAL EXAMPLE

Consider a one dimensional stochastic process characterized by the iteration

$$v_{t+1} = (1-\gamma_t)v_t + \gamma_t(K_t(v_t) + w_t), \tag{7}$$

where γ_t is the learning rate and w_t is a noise term. Let us suppose we have n alternating operators k_i with Lipschitz constants $b_i < 1$ and fixed points v_i^* where $i \in \{0,\ldots,n-1\}$,

$$k_i(v) = v + (1-b_i)(v_i^* - v).$$

The current operator at time t is $K_t = k_i$ (thus, $V_t^* = v_i^*$ and $\beta_t = b_i$) if $i \equiv t \pmod n$, that is, if i is congruent with t modulo n: if they have the same remainder when they are divided by n. In other words, we apply a *round-robin* type schedule for the operators.

Figure 1 shows that the trajectories remained close to the fixed points. The figure illustrates the case of two (-1 and 1) and six ($-3,-2,-1,1,2,3$) alternating fixed points.

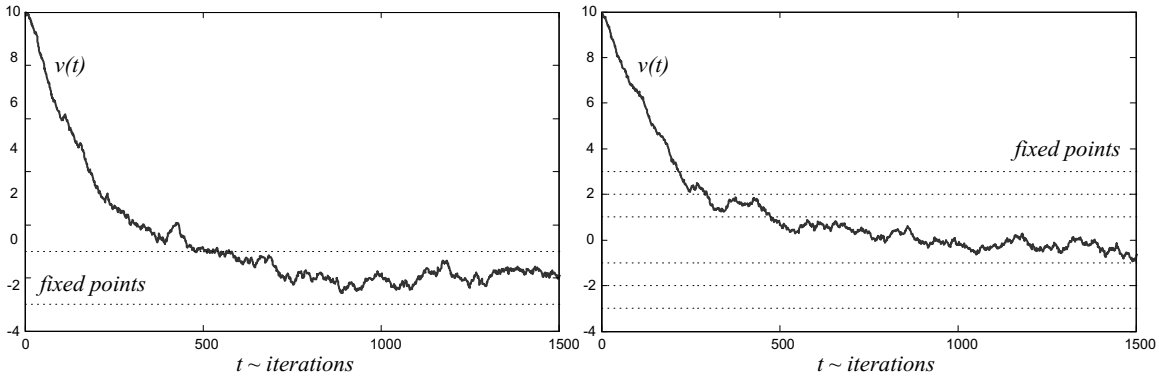


Figure 1: Trajectories generated by (7) with two (left) and six (right) fixed points.

5.2.2 A PATHOLOGICAL EXAMPLE

During this example we will restrict ourselves to deterministic functions. According to the *Banach fixed point theorem*, if we have a contraction mapping f over a complete metric space with fixed point $v^* = f(v^*)$, then, for any initial v_0 the sequence $v_{t+1} = f(v_t)$ converges to v^*. It could be thought that this result can be easily generalized to the case of alternating operators. For example, suppose we have n alternating contraction mappings k_i with Lipschitz constants $b_i < 1$ and fixed points v_i^*, respectively, where $i \in \{0,\ldots,n-1\}$, and we apply them iteratively starting from an arbitrary v_0, viz., $v_{t+1} = K_t(v_t)$, where $K_t = k_i$ if $i \equiv t \pmod n$. One may think that since each k_i attracts the point towards its fixed point, the sequence v_t converges to the *convex hull* of the fixed points. However, as the following example demonstrates, this is not the case, since it is possible that the point moves away from the convex hull and, in fact, it gets farther and farther after each iteration.

Now, let us consider two one-dimensional functions, $k_i : \mathbb{R} \to \mathbb{R}$, where $i \in \{a,b\}$, defined below by Equation (8). It can be easily proven that these functions are contractions with fixed points v_i^*

and Lipschitz constants b_i (in Figure 2, $v_a^* = 1$, $v_b^* = -1$ and $b_i = 0.9$).

$$k_i(v) = \begin{cases} v + (1-b_i)(v_i^* - v) & \text{if } sgn(v_i^*) = sgn(v - v_i^*), \\ v_i^* + (v_i^* - v) + (1-b_i)(v - v_i^*) & \text{otherwise,} \end{cases} \tag{8}$$

where $sgn(\cdot)$ denotes the signum[1] function. Figure 2 demonstrates that even if the iteration starts from the middle of the convex hull (from the center of mass), $v_0 = 0$, it starts getting farther and farther from the fixed points in each step when we apply k_a and k_b after each other. Nevertheless,

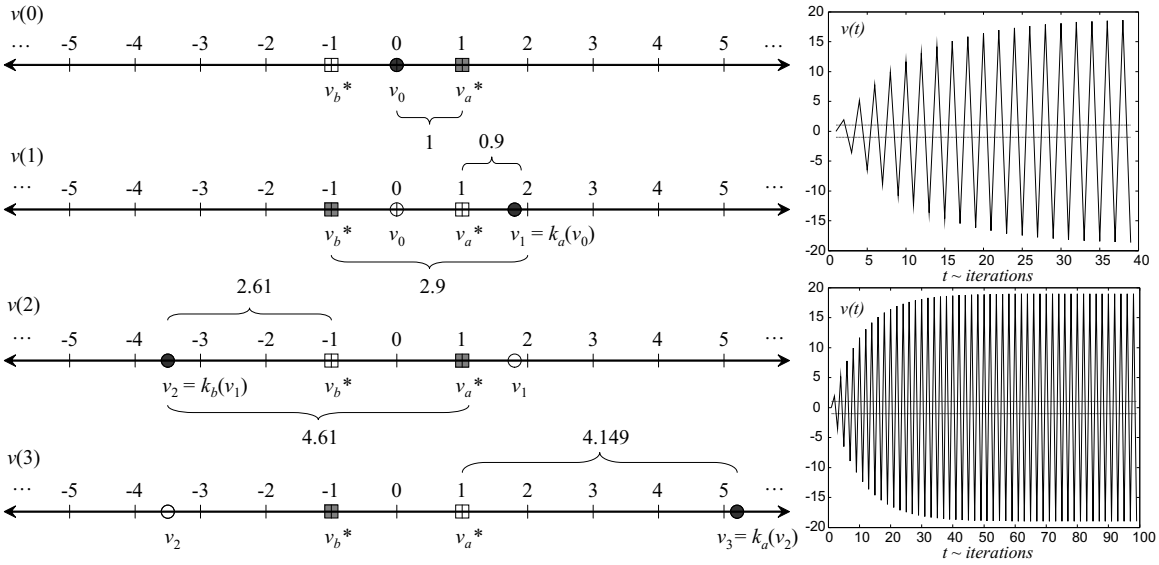


Figure 2: A deterministic pathological example, generated by the iterative application of (8). The left part demonstrates the first steps, while the two images on the right-hand side show the behavior of the trajectory in the long run.

the following argument shows that sequence v_t cannot get arbitrarily far from the fixed points. Let us denote the *diameter* of the convex hull of the fixed points by ρ. Since this convex hull is a polygon (where the vertices are fixed points) $\rho = \max_{i,j} \|v_i^* - v_j^*\|$. Furthermore, let β_0 be defined as $\beta_0 = \max_i b_i$ and d_t as $d_t = \min_i \|v_i^* - v_t\|$. Then, it can be proven that for all t, we have $d_{t+1} \leq \beta_0(2\rho + d_t)$. If we assume that $d_{t+1} \geq d_t$, then it follows that $d_t \leq d_{t+1} \leq \beta_0(2\rho + d_t)$. After rearrangement, we get the following inequality

$$d_t \leq \frac{2\beta_0\rho}{1-\beta_0} = \phi(\beta_0, \rho).$$

Therefore, $d_t > \phi(\beta_0, \rho)$ implies that $d_{t+1} < d_t$. Consequently, if v_t somehow got farther than $\phi(\beta_0, \rho)$, in the next step it would inevitably be attracted towards the fixed points. It is easy to see that this argument is valid in an arbitrary normed space, as well.

1. $sgn(x) = 0$ if $x = 0$, $sgn(x) = -1$ if $x < 0$ and $sgn(x) = 1$ if $x > 0$.

5.3 Reinforcement Learning in (ε,δ)-MDPs

In case of finite (ε,δ)-MDPs we can formulate a relaxed convergence theorem for value function based reinforcement learning algorithms, as a corollary of Theorem 20. Suppose that $\mathcal{V}$ consists of state-value functions, namely, $\mathcal{X}=\mathbb{X}$. Then, we have

$$\limsup_{t\to\infty}\|J^*-J_t^*\|_\infty \le d(\varepsilon,\delta),$$

where J_t^* is the optimal value function of the MDP at time t and J^* is the optimal value function of the base MDP. In order to calculate $d(\varepsilon,\delta)$, Theorems 11 (or 10), 12 and the triangle inequality could be applied. Assume, for example, that we use the supremum norm, $\|\cdot\|_\infty$, for cost functions and $\|\cdot\|_1$, defined by Equation (5), for transition functions. Then,

$$d(\varepsilon,\delta)=\frac{\varepsilon\alpha\|g\|_\infty}{(1-\alpha)^2}+\frac{\delta}{1-\alpha},$$

where g is the cost function of the base MDP. Now, by applying Theorem 20, we have

Corollary 21 *Suppose that we have an (ε,δ)-MDP and Assumptions 1-3 hold. Let V_t be the sequence generated by iteration (6). Furthermore, assume that the fixed point of each operator K_t is J_t^*. Then, for any initial $V_0\in\mathcal{V}$, the sequence V_t κ-approximates J^* with*

$$\kappa=\frac{4d(\varepsilon,\delta)}{1-\beta_0}.$$

Notice that as parameters ε and δ go to zero, we get back to a classical convergence theorem for this kind of stochastic iterative algorithm (still in a little bit generalized form, since β_t might still change over time). Now, with the help of these results, we will investigate the convergence of some classical reinforcement learning algorithms in (ε,δ)-MDPs.

5.3.1 ASYNCHRONOUS VALUE ITERATION IN (ε,δ)-MDPS

The method of value iteration is one of the simplest reinforcement learning algorithms. In ordinary MDPs it is defined by the iteration $J_{t+1}=TJ_t$, where T is the Bellman operator. It is known that the sequence J_t converges in the supremum norm to J^* for any initial J_0 (Bertsekas and Tsitsiklis, 1996). The asynchronous variant of value iteration arises when the states are updated asynchronously, for example, only one state in each iteration. In the case of (ε,δ)-MDPs a small stepsize variant of asynchronous value iteration can be defined as

$$J_{t+1}(x)=(1-\gamma_t(x))J_t(x)+\gamma_t(x)(T_tJ_t)(x),$$

where T_t is the Bellman operator of the current MDP at time t. Since there is no noise term in the iteration, Assumption 1 is trivially satisfied. Assumption 3 follows from the fact that each T_t operator is an α contraction where α is the discount factor. Therefore, if the stepsizes satisfy Assumption 2 then, by applying Corollary 21, we have that the sequence J_t κ-approximates J^* for any initial value function J_0 with $\kappa=(4d(\varepsilon,\delta))/(1-\alpha)$.

5.3.2 Q-LEARNING IN (ε,δ)-MDPS

Watkins' Q-learning is a very popular off-policy model-free reinforcement learning algorithm (Even-Dar and Mansour, 2003). Its generalized version in ε-MDPs was studied by Szita et al. (2002). The Q-learning algorithm works with action-value functions, therefore, $\mathcal{X} = \mathbb{X} \times \mathbb{A}$, and the one-step Q-learning rule in (ε,δ)-MDPs can be defined as follows

$$Q_{t+1}(x,a) = (1-\gamma_t(x,a))Q_t(x,a) + \gamma_t(x,a)(\widetilde{T}_t Q_t)(x,a), \tag{9}$$

$$(\widetilde{T}_t Q_t)(x,a) = g_t(x,a) + \alpha \min_{B\in\mathcal{A}(Y)} Q_t(Y,B),$$

where g_t is the immediate-cost function of the current MDP at time t and Y is a random variable generated from the pair (x,a) by simulation, that is, according to the probability distribution $p_t(x,a)$, where p_t is the transition function of the current MDP at time t.

Operator $\widetilde{T}_t$ is randomized, but as it was shown by Bertsekas and Tsitsiklis (1996) in their convergence theorem for Q-learning, it can be rewritten in a form as follows

$$(\widetilde{T}_t Q)(x,a) = (\widetilde{K}_t Q)(x,a) + \widetilde{W_t}(x,a),$$

where $\widetilde{W_t}(x,a)$ is a noise term with zero mean and finite variance, and $\widetilde{K}_t$ is defined as

$$(\widetilde{K}_t Q)(x,a) = g_t(x,a) + \alpha \sum_{y\in\mathbb{X}} p_t(y \mid x,a) \min_{b\in\mathcal{A}(y)} Q(y,b).$$

Let us denote the optimal action-value function of the current MDP at time t and the base MDP by Q_t^* and Q^*, respectively. By using the fact that $J^*(x) = \min_a Q^*(x,a)$, it is easy to see that for all t, Q_t^* is the fixed point of operator $\widetilde{K}_t$ and, moreover, each $\widetilde{K}_t$ is an α contraction. Therefore, if the stepsizes satisfy Assumption 2, then the Q_t sequence generated by iteration (9) κ-approximates Q^* for any initial Q_0 with $\kappa = (4d(\varepsilon,\delta))/(1-\alpha)$.

In some situations the immediate costs are randomized, however, even in this case the relaxed convergence of Q-learning would follow as long as the random immediate costs had finite expected value and variance, which is required for satisfying Assumption 1.

5.3.3 TEMPORAL DIFFERENCE LEARNING IN (ε,δ)-MDPS

Temporal difference learning, or for short TD-learning, is a policy evaluation algorithm. It aims at finding the corresponding value function J^π for a given control policy π (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998). It can also be used for approximating the optimal value function, for example, if we apply it together with the policy iteration algorithm.

First, we briefly review the off-line first-visit variant of TD(λ) in case of ordinary MDPs. It can be shown that the value function of a policy π can be rewritten in a form as

$$J^\pi(x) = \mathbb{E}\left[\sum_{m=0}^{\infty} (\alpha\lambda)^m D^\pi_{\alpha,m} \,\middle|\, X_0 = x\right] + J^\pi(x),$$

where $\lambda \in [0,1)$ and $D^\pi_{\alpha,m}$ denotes the "temporal difference" coefficient at time m,

$$D^\pi_{\alpha,m} = g(X_m, A^\pi_m) + \alpha J^\pi(X_{m+1}) - J^\pi(X_m),$$

where X_m, X_{m+1} and A_m^π are random variables, X_{m+1} has $p(X_m, A_m^\pi)$ distribution and A_m^π is a random variable for actions, it is selected according to the distribution $\pi(X_m)$.

Based on this observation, we can define a stochastic approximation algorithm as follows. Let us suppose that we have a generative model of the environment, for example, we can perform simulations in it. Each simulation produces a state-action-reward trajectory. We can assume that all simulations eventually end, for example, there is an absorbing termination state or we can stop the simulation after a given number of steps. Note that even in this case we can treat each trajectory as infinitely long, viz., we can define all costs after the termination as zero. The off-line first-visit TD(λ) algorithm updates the value function after each simulation,

$$J_{t+1}(x_k^t) = J_t(x_k^t) + \gamma_t(x_k^t) \sum_{m=k}^{\infty} (\alpha\lambda)^{m-k} d_{\alpha,m,t}, \tag{10}$$

where x_k^t is the state at step k in trajectory t and $d_{\alpha,m,t}$ is the temporal difference coefficient,

$$d_{\alpha,m,t} = g(x_m^t, a_m^t) + \alpha J_t(x_{m+1}^t) - J_t(x_m^t).$$

For the case of ordinary MDPs it is known that TD(λ) converges almost surely to J^π for any initial J_0 provided that each state is visited by infinitely many trajectories and the stepsizes satisfy Assumption 2. The proof is based on the observation that iteration (10) can be seen as a Robbins-Monro type stochastic iterative algorithm for finding the fixed point of $J^\pi = HJ^\pi$, where H is a contraction mapping with Lipschitz constant α (Bertsekas and Tsitsiklis, 1996). The only difference in the case of (ε, δ)-MDPs is that the environment may change over time and, therefore, operator H becomes time-dependent. However, each H_t is still an α contraction, but they potentially have different fixed points. Therefore, we can apply Theorem 20 to achieve a relaxed convergence result for off-line first-visit TD(λ) in changing environments under the same conditions as in the case of ordinary MDPs.

The convergence of the on-line every-visit variant can be proven in the same way as in the case of ordinary MDPs, viz., by showing that the difference between the two variants is of second order in the size of γ_t and hence inconsequential as γ_t diminishes to zero.

5.3.4 APPROXIMATE DYNAMIC PROGRAMMING

Most RL algorithms in their standard forms, for example, with lookup table representations, are highly intractable in practice. This phenomenon, which was named "curse of dimensionality" by Bellman, has motivated approximate approaches that result in more tractable methods, but often yield suboptimal solutions. These techniques are usually referred to as *approximate dynamic programming* (ADP). Many ADP methods are combined with simulation, but their key issue is to approximate the value function with a suitable *approximation architecture*: $V \approx \Phi(r)$, where r is a parameter vector. Direct ADP methods collect samples by using simulation, and fit the architecture to the samples. Indirect methods obtain parameter r by using an approximate version of the Bellman equation (Bertsekas, 2007).

The *power of the approximation architecture* is the smallest error that can be achieved, $\eta = \inf_r \|V^* - \Phi(r)\|$, where V^* is the optimal value function. Suppose that $\eta > 0$, then no algorithm can provide a result whose distance from V^* is less than η. Hence, the maximum that we can hope for is to converge to an environment of V^* (Bertsekas and Tsitsiklis, 1996). In what follows, we briefly investigate the connection of our results with ADP.

In general, many direct and indirect ADP methods can be formulated as follows

$$\Phi(r_{t+1}) = \Pi\big((1-\gamma_t)\Phi(r_t) + \gamma_t(B_t(\Phi(r_t)) + W_t)\big), \tag{11}$$

where $r_t \in \Theta$ is an approximation parameter, Θ is the parameter space, for example, $\Theta \subseteq \mathbb{R}^p$, $\Phi : \Theta \to \mathcal{F}$ is an approximation architecture where $\mathcal{F} \subseteq \mathcal{V}$ is a Hilbert space that can be represented by using Φ with parameters from Θ. Function $\Pi : \mathcal{V} \to \mathcal{F}$ is a projection mapping, it renders a representation from $\mathcal{F}$ to each value function from $\mathcal{V}$. Operator $B_t : \mathcal{F} \to \mathcal{V}$ acts on (approximated) value functions. Finally, γ_t denotes the stepsize and W_t is a noise parameter representing the uncertainties coming from, for example, the simulation.

Operator B_t is time-dependent since, for example, if we model an approximate version of optimistic policy iteration, then in each iteration the control policy changes and, therefore, the update operator changes, as well. We can notice that if Π was a linear operator (see below), Equation (11) would be a stochastic iterative algorithm with $K_t = \Pi B_t$. Consequently, the algorithm described by Equation (6) is a generalization of many ADP methods, as well.

Now, we show that a convergence theorem for ADP methods can also be deduced by using Theorem 20. In order to apply the theorem, we should ensure that each update operator be a contraction. If we assume that every B_t is a contraction, we should require two properties from Π to guarantee that the resulted operators remain contractions. First, Π should be *linear*. Operator Π is linear if it is *additive* and *homogeneous*, more precisely, if $\forall V_1, V_2 : \Pi(V_1 + V_2) = \Pi(V_1) + \Pi(V_2)$ and $\forall V : \forall \alpha : \Pi(\alpha V) = \alpha \Pi(V)$, where α is a scalar. This requirement allows the separation of the components. Moreover, Π should be *nonexpansive* w.r.t. the supremum norm, namely: $\forall V_1, V_2 : \|\Pi(V_1) - \Pi(V_2)\| \leq \|V_1 - V_2\|$. Then, the update operator of the algorithm, $K_t = \Pi B_t$, is guaranteed to be a contraction.

If we assume that V_t^* is the fixed point of K_t, thus, $(\Pi B_t)V_t^* = V_t^*$ and β_t is the Lipschitz constant of K_t with $\limsup_{t\to\infty} \beta_t = \beta_0 < 1$, we can deduce a convergence theorem for ADP methods, as a corollary of Theorem 20. Suppose that Assumptions 1-2 hold and each B_t is a contraction as well as Π is linear and supremum norm nonexpansive, then $\Phi(r_t)$ κ-approximates V^* for any initial r_0 with $\kappa = 4\rho/(1-\beta_0)$, where $\rho = \limsup_{t\to\infty} \|V_t^* - V^*\|$. In case all of the fixed points were the same, viz., $\forall t : V_0^* = V_t^*$, then $\Phi(r_t)$ would converge to V_0^* almost surely, consequently, $\Phi(r_t)$ would κ-approximate V^* with $\kappa = \|V_0^* - V^*\|$.

Naturally, these results are quite loose, since we did not make strong assumptions on the applied algorithm and on the approximation architecture. They only illustrate that the approach we took, which allows time-dependent update operators and analyzes approximate convergence, could also provide results for ordinary MDPs, for example, in the case of ADP.

6. Experimental Results

In this section we present two numerical experiments. The first one demonstrates the effects of environmental changes during Q-learning based *scheduling*. The second one presents a parameter analysis concerning the effectiveness of SARSA in (ε, δ)-type *grid world* domains.

6.1 Environmental Changes During Scheduling

Scheduling is the allocation of *resources* over time to perform a collection of *jobs*. Each job consists of a set of *tasks*, potentially with precedence constraints, to be executed on the resources. The

job-shop scheduling problem (JSP) is one of the basic scheduling problems (Pinedo, 2002). We investigated an extension of JSP, called the *flexible job-shop* scheduling problem (FJSP), in which some of the resources are interchangeable, that is, there may be tasks that can be executed on several resources. This problem can be formulated as a finite horizon MDP and can be solved by Q-learning based methods (Csáji and Monostori, 2006).

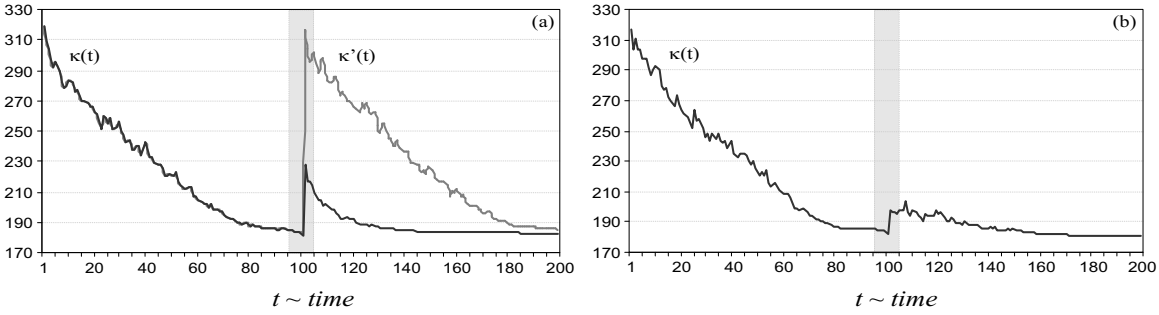


Figure 3: The black curves, $\kappa(t)$, show the performance measure in case there was a resource breakdown (a) or a new resource availability (b) at time $t = 100$; the gray curve in (a), $\kappa'(t)$, demonstrates the case the policy would be recomputed from scratch.

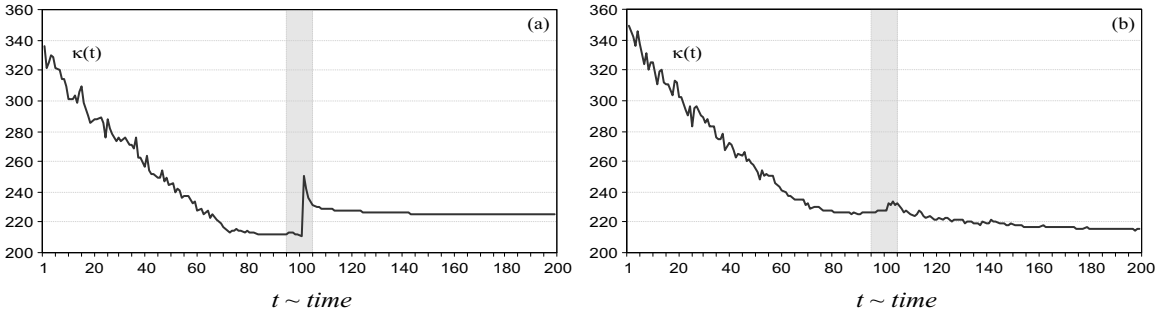


Figure 4: The black curves, $\kappa(t)$, show the performance measure during resource control in case there was a new job arrival (a) or a job cancellation (b) at time $t = 100$.

In order to investigate the effects of environmental changes during scheduling, numerical experiments were initiated and carried out. The aim of scheduling was to minimize the maximum completion time of the tasks, which performance measure is called "makespan". The adaptive features of the Q-learning based approach were tested by confronting the system with unexpected events, such as: resource breakdown, new resource availability (Figure 3), new job arrival or job cancellation (Figure 4). In Figures 3 and 4 the horizontal axis represents time, while the vertical one, the achieved performance measure. The figures were made by averaging hundred random samples. In these tests a fixed number of 20 resources were used with few dozens of jobs, where each job contained a sequence of tasks. In each case there was an unexpected event at time $t = 100$. After the change took place, we considered two possibilities: we either restarted the iterative scheduling process from scratch or we continued the learning using the current (obsolete) value function. We experienced that the latter approach is much more efficient. That was one of the reasons why we started to study how the optimal value function of an MDP depends on the dynamics of the system.

Recall that Theorems 10, 11 and 12 measure the amount of the possible change in the value function in case there were changes in the MDP, but since these theorems apply supremum norm, they only provide bounds for *worst case* situations. However, the results of our numerical experiments, shown in Figures 3 and 4, are indicative of the phenomenon that in an *average case* the change is much less. Therefore, applying the obsolete value function after a change took place is preferable over restarting the optimization from scratch.

The results, black curves, show the case when the obsolete value function approximation was applied after the change took place. The performance which would arise if the system recomputed the whole schedule from scratch is drawn in gray in part (a) of Figure 3.

6.2 Varying Grid World

We also performed numerical experiments on a variant of the classical *grid world* problem (Sutton and Barto, 1998). The original version of this problem can be briefly described as follows: an agent wanders in a rectangular world starting from a random *initial* state with the aim of finding the *goal* state. In each state the agent is allowed to choose from four possible actions: "north", "south", "east" and "west". After an action was selected, the agent moves one step in that direction. There are some *mines* on the field, as well, that the agent should avoid. An *episode* ends if the agent finds the goal state or hits a mine. During our experiments, we applied randomly generated 10×10 grid worlds (thus, these MDPs had 100 states) with 10 mines. The *immediate-cost* of taking a (non-terminating) step was 5, a cost of hitting a mine was 100 and the cost of finding the goal state was -100.

In order to perform the experiment described by Table 1, we have applied the "RL-Glue" framework[2] which consists of open source softwares and aims at being a standard protocol for benchmarking and interconnecting reinforcement learning agents and environments.

We have analyzed an (ε, δ)-type version of grid world, where the problem formed an (ε, δ)-MDP. More precisely, we have investigated the case when for all time t, the transition-probabilities could vary by at most $\varepsilon \geq 0$ around the base transition-probability values and the immediate-costs could vary by at most $\delta \geq 0$ around the base cost values.

During our numerical experiments, the environment changed at each time-step. These changes were generated as follows. First, changes concerning the transition-probabilities are described. In our randomized grid worlds the agent was taken to a random surrounding state (no matter what action it chose) with probability η and this probability *changed* after each step. The new η was computed according to the *uniform* distribution, but its possible values were *bounded* by the values described in the first row of Table 1.

Similarly, the immediate-costs of the base MDP (cf. the first paragraph) were *perturbed* with a uniform random variable that changed at each time-step. Again, its (absolute) value was bounded by δ, which is presented in the first column of the table. The values shown were divided by 100 to achieve the same scale as the transition-probabilities have.

Table 1 was generated using an (optimistic) SARSA algorithm, namely, the current policy was evaluated by SARSA, then the policy was (optimistically) improved, more precisely, the *greedy* policy with respect to the achieved evaluation was calculated. That policy was also *soft*, namely, it made random *explorations* with probability 0.05. We have generated 1000 random grid worlds for each parameter pairs and performed 10 000 episodes in each of these generated worlds. The results

2. RL-Glue can be found at http://rlai.cs.ualberta.ca/RLBB/top.html.

$\Delta\|g\|$	the bounds for the varying probability of arriving at random states $\sim \varepsilon$										
$\delta/100$	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
0.0	-55.5	-48.8	-41.4	-36.7	-26.7	-16.7	-8.5	2.1	14.2	31.7	46.0
0.1	-54.1	-46.1	-41.2	-34.5	-25.8	-15.8	-6.0	3.7	16.5	32.3	46.3
0.2	-52.5	-44.8	-40.1	-34.4	-25.3	-15.4	-5.8	4.0	17.6	33.1	48.1
0.3	-49.7	-42.1	-36.3	-31.3	-23.9	-14.2	-5.3	8.0	18.1	37.2	51.6
0.4	-47.4	-41.5	-34.7	-30.7	-22.2	-12.2	-2.3	8.8	20.2	38.3	52.0
0.5	-42.7	-41.0	-34.5	-24.8	-21.1	-10.1	-1.3	11.2	25.7	39.2	52.1
0.6	-36.1	-36.5	-29.7	-24.0	-16.8	-7.9	1.1	17.0	31.3	43.9	54.1
0.7	-30.2	-29.3	-29.3	-19.1	-13.4	-6.0	7.4	18.9	26.9	47.2	60.9
0.8	-23.1	-27.0	-21.4	-18.8	-10.9	-2.6	8.9	22.5	31.3	50.0	64.2
0.9	-14.1	-19.5	-21.0	-12.4	-7.5	0.7	13.2	23.2	38.9	52.2	68.1
1.0	-6.8	-10.7	-14.5	-7.1	-5.3	6.6	15.7	26.4	39.8	57.3	68.7

Table 1: The (average) cumulative costs gathered by SARSA in varying grid worlds.

presented in the table were calculated by averaging the cumulative costs over all episodes and over all generated sample worlds.

The parameter analysis shown in Table 1 is indicative of the phenomenon that changes in the transition-probabilities have a much higher impact on the performance. Even large perturbations in the costs were tolerated by SARSA, but large variations in the transition-probabilities caused a high decrease in the performance. An explanation could be that large changes in the transitions cause the agent to loose control over the events, since it becomes very hard to predict the effects of the actions and, hence, to estimate the expected costs.

7. Conclusion

The theory of MDPs provide a general framework for modeling decision making in stochastic dynamic systems, if we know a function that describes the dynamics or we can simulate it, for example, with a suitable program. In some situations, however, the dynamics of the system may change, too. In theory, this change can be modeled with another (higher level) MDP, as well, but doing so would lead to models which are practically intractable.

In the paper we have argued that the optimal value function of a (discounted) MDP Lipschitz continuously depends on the transition-probability function and the immediate-cost function, therefore, small changes in the environment result only in small changes in the optimal value function. This result was already known for the case of transition-probabilities, but we have presented an improved estimation for this case, as well. A bound for changes in the discount factor was also proven, and it was demonstrated that, in general, this dependence was not Lipschitz continuous. Additionally, it was shown that changes in the discount rate could be traced back to changes in the immediate-cost function. The application of the Lipschitz property helps the theoretical treatment of changing environments or inaccurate models, for example, if the transition-probabilities or the costs are estimated statistically, only.

In order to theoretically analyze environmental changes, the framework of (ε,δ)-MDPs was introduced as a generalization of classical MDPs and ε-MDPs. In this quasi-stationary model the

transition-probability function and the immediate-cost function may change over time, but the cumulative changes must remain bounded by ε and δ, asymptotically.

Afterwards, we have investigated how RL methods could work in this kind of changing environment. We have presented a general theorem that estimated the asymptotic distance of a value function sequence from a fixed value function. This result was applied to deduce a convergence theorem for value function based algorithms that work in (ε, δ)-MDPs.

In order to demonstrate our approach, we have presented some numerical experiments, too. First, two simple iterative processes were shown, a "well-behaving" stochastic process and a "pathological", oscillating deterministic process. Later, the effects of environmental changes on Q-learning based flexible job-shop scheduling was experimentally studied. Finally, we have analyzed how SARSA could work in varying (ε, δ)-type grid world domains.

We can conclude that value function based RL algorithms can work in varying environments, at least if the changes remain bounded in the limit. The asymptotic distance of the generated value function sequence from the optimal value function of the base MDP is bounded for a large class of stochastic iterative algorithms. Moreover, this bound is proportional to the diameter of this set, for example, to parameters ε and δ in the case of (ε, δ)-MDPs. These results were illustrated through three classical RL methods: asynchronous value iteration, Q-learning and temporal difference learning policy evaluation. We showed, as well, that this approach could be applied to investigate the convergence of ADP methods.

There are many potential further research directions. Now, as a conclusion to the paper, we highlight some of them. First, analyzing the effects of environmental changes on the value function in case of the *expected average cost* optimization criterion would be interesting. A promising direction could be to investigate environments with non-bounded changes, for example, when the environment might *drift* over time. Naturally, this drift should also be sufficiently slow in order to give the opportunity to the learning algorithm to *track* the changes. Another possible direction could be the further analysis of the convergence results in case of applying *value function approximation*. The classical problem of *exploration* and *exploitation* should also be reinvestigated in changing environments. Finally, for practical reasons, it would be important to find *finite time bounds* for the convergence of stochastic iterative algorithms for (a potentially restricted class of) non-stationary environments.

Acknowledgments

The work was supported by the Hungarian Scientific Research Fund (OTKA), Grant No. T73376, and by the EU-project Coll-Plexity, 12781 (NEST). Balázs Csanád Csáji greatly acknowledges the scholarship of the Hungarian Academy of Sciences. The authors are also very grateful to Csaba Szepesvári for the helpful comments and discussions.

Appendix A. Proofs

In this appendix the proofs of Theorems 11, 12, 13, 20 and Lemmas 14, 18 can be found.

Theorem 11 *Assume that two MDPs differ only in their transition-probability functions, denoted by p_1 and p_2. Let the corresponding optimal value functions be J_1^* and J_2^*, then*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{\alpha \|g\|_\infty}{(1-\alpha)^2} \|p_1 - p_2\|_1 ,$$

where $\|\cdot\|_1$ is a norm on $f : \mathbb{X} \times \mathbb{A} \times \mathbb{X} \to \mathbb{R}$ type functions, for example, $f(x,a,y) = p(y|x,a)$,

$$\|f\|_1 = \max_{x,a} \sum_{y \in \mathbb{X}} |f(x,a,y)| .$$

Proof First, let us introduce a deterministic Markovian policy. For all state $x \in \mathbb{X}$:

$$\hat{\pi}(x) = \begin{cases} \arg\min\limits_{a \in \mathcal{A}(x)} \left[g(x,a) + \alpha \sum\limits_{y \in \mathbb{X}} p_1(y \mid x,a) J_1^*(y) \right] & \text{if } J_1^*(x) \leq J_2^*(x), \\ \arg\min\limits_{a \in \mathcal{A}(x)} \left[g(x,a) + \alpha \sum\limits_{y \in \mathbb{X}} p_2(y \mid x,a) J_2^*(y) \right] & \text{if } J_2^*(x) < J_1^*(x) \end{cases}$$

If the arg min is ambiguous then any action that takes the minimum can be selected. Using the Bellman optimality equation in the first step, $\|J_1^* - J_2^*\|_\infty$ can be estimated as follows,

$$\forall x \in \mathbb{X} : |J_1^*(x) - J_2^*(x)| =$$

$$= \left| \min_{a \in \mathcal{A}(x)} \left[g(x,a) + \alpha \sum_{y \in \mathbb{X}} p_1(y \mid x,a) J_1^*(y) \right] - \min_{a \in \mathcal{A}(x)} \left[g(x,a) + \alpha \sum_{y \in \mathbb{X}} p_2(y \mid x,a) J_2^*(y) \right] \right| \leq$$

$$\leq \left| g(x,\hat{\pi}(x)) + \alpha \sum_{y \in \mathbb{X}} p_1(y \mid x,\hat{\pi}(x)) J_1^*(y) - g(x,\hat{\pi}(x)) - \alpha \sum_{y \in \mathbb{X}} p_2(y \mid x,\hat{\pi}(x)) J_2^*(y) \right| ,$$

where we applied that $\forall f_1, f_2 : \mathcal{S} \to \mathbb{R}$ bounded functions such that $\min_s f_1(s) \leq \min_s f_2(s)$ and $\hat{s} = \arg\min_s f_1(s)$, we have $|\min_s f_1(s) - \min_s f_2(s)| \leq |f_1(\hat{s}) - f_2(\hat{s})|$. Then,

$$\forall x \in \mathbb{X} : |J_1^*(x) - J_2^*(x)| \leq \left| \alpha \sum_{y \in \mathbb{X}} p_1(y \mid x,\hat{\pi}(x)) J_1^*(y) - p_2(y \mid x,\hat{\pi}(x)) J_2^*(y) \right| =$$

$$= \left| \alpha \sum_{y \in \mathbb{X}} (p_1(y \mid x,\hat{\pi}(x)) - p_2(y \mid x,\hat{\pi}(x))) J_1^*(y) + \alpha \sum_{y \in \mathbb{X}} p_2(y \mid x,\hat{\pi}(x)) (J_1^*(y) - J_2^*(y)) \right| \leq$$

$$\leq \alpha \sum_{y \in \mathbb{X}} |(p_1(y \mid x,\hat{\pi}(x)) - p_2(y \mid x,\hat{\pi}(x))) J_1^*(y)| + \alpha \sum_{y \in \mathbb{X}} |p_2(y \mid x,\hat{\pi}(x)) (J_1^*(y) - J_2^*(y))| ,$$

where in the second step we have rewritten $p_1(y|x,\hat{\pi}(x)) J_1^*(y) - p_2(y|x,\hat{\pi}(x)) J_2^*(y)$ as

$$p_1(y|x,\hat{\pi}(x)) J_1^*(y) - p_2(y|x,\hat{\pi}(x)) J_2^*(y) =$$

$$= p_1(y|x,\hat{\pi}(x)) J_1^*(y) - p_2(y|x,\hat{\pi}(x)) J_1^*(y) + p_2(y|x,\hat{\pi}(x)) J_1^*(y) - p_2(y|x,\hat{\pi}(x)) J_2^*(y) =$$

$$= (p_1(y\,|\,x,\hat{\pi}(x)) - p_2(y\,|\,x,\hat{\pi}(x)))J_1^*(y) + p_2(y\,|\,x,\hat{\pi}(x))(J_1^*(y) - J_2^*(y)).$$

Now, let us recall (a special form of) *Hölder's inequality*: let v_1, v_2 be two vectors and $1 \leq q, r \leq \infty$ with $1/q + 1/r = 1$. Then, we have $\|v_1 v_2\|_{(1)} \leq \|v_1\|_{(q)} \|v_2\|_{(r)}$, where $\|\cdot\|_{(q)}$ denotes *vector* norm, for example, $\|v\|_{(q)} = (\sum_i |v_i|^q)^{1/q}$ and $\|v\|_{(\infty)} = \max_i |v_i| = \|v\|_\infty$. Here, we applied the unusual "(q)" notation to avoid confusion with the applied matrix norm. Notice that the first sum of the last estimation can be treated as the (1)-norm of $v_1 v_2$, where

$$v_1(y) = p_1(y \mid x,\hat{\pi}(x)) - p_2(y \mid x,\hat{\pi}(x))) \quad \text{and} \quad v_2(y) = J_1^*(y),$$

after which Hölder's inequality can be applied with $q = 1$ and $r = \infty$ to estimate the sum. A similar argument can be repeated in the case of the second sum with

$$v_1(y) = p_2(y \mid x,\hat{\pi}(x)) \quad \text{and} \quad v_2(y) = J_1^*(y) - J_2^*(y).$$

Then, after the two applications of Hölder's inequality, we have for all x that

$$|J_1^*(x) - J_2^*(x)| \leq \alpha \|p_1(\cdot \mid x,\hat{\pi}(x)) - p_2(\cdot \mid x,\hat{\pi}(x))\|_{(1)} \|J_1^*\|_\infty +$$

$$+ \alpha \|p_2(\cdot \mid x,\hat{\pi}(x))\|_{(1)} \|J_1^* - J_2^*\|_\infty,$$

since $\|J_1^*\|_\infty \leq \|g\|_\infty/(1-\alpha)$, $\|p_2(\cdot \mid x,\hat{\pi}(x))\|_{(1)} = 1$ and we have this estimation for all x,

$$\|J_1^* - J_2^*\|_\infty \leq \frac{\alpha\|g\|_\infty}{1-\alpha} \max_{x\in\mathbb{X}} \sum_{y\in\mathbb{X}} |\,p_1(y\,|\,x,\hat{\pi}(x)) - p_2(y\,|\,x,\hat{\pi}(x))\,| + \alpha\|J_1^* - J_2^*\|_\infty,$$

which formula can be overestimated, by taking the maximum over all actions, by

$$\|J_1^* - J_2^*\|_\infty \leq \frac{\alpha\|g\|_\infty}{1-\alpha} \|p_1 - p_2\|_1 + \alpha\|J_1^* - J_2^*\|_\infty,$$

from which the statement of the theorem immediately follows after rearrangement. ■

Theorem 12 *Assume that two discounted MDPs differ only in the immediate-cost functions, denoted by g_1 and g_2. Let the corresponding optimal value functions be J_1^* and J_2^*, then*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{1}{1-\alpha} \|g_1 - g_2\|_\infty.$$

Proof First, let us introduce a deterministic Markovian policy. For all state $x \in \mathbb{X}$:

$$\hat{\pi}(x) = \begin{cases} \displaystyle\operatorname*{arg\,min}_{a\in\mathcal{A}(x)} \left[g_1(x,a) + \alpha \sum_{y\in\mathbb{X}} p(y \mid x,a) J_1^*(y) \right] & \text{if } J_1^*(x) \leq J_2^*(x), \\ \displaystyle\operatorname*{arg\,min}_{a\in\mathcal{A}(x)} \left[g_2(x,a) + \alpha \sum_{y\in\mathbb{X}} p(y \mid x,a) J_2^*(y) \right] & \text{if } J_2^*(x) < J_1^*(x). \end{cases}$$

If the arg min is ambiguous, then any action that takes the minimum can be selected. Using the Bellman optimality equation in the first step, $\|J_1^* - J_2^*\|_\infty$ can be estimated as follows,

$$\forall x \in \mathbb{X} : |J_1^*(x) - J_2^*(x)| =$$

$$= \left| \min_{a \in \mathcal{A}(x)} \left[g_1(x,a) + \alpha \sum_{y \in \mathbb{X}} p(y \mid x,a) J_1^*(y) \right] - \min_{a \in \mathcal{A}(x)} \left[g_2(x,a) + \alpha \sum_{y \in \mathbb{X}} p(y \mid x,a) J_2^*(y) \right] \right| \leq$$

$$\leq \left| g_1(x,\hat{\pi}(x)) + \alpha \sum_{y \in \mathbb{X}} p(y \mid x,\hat{\pi}(x)) J_1^*(y) - g_2(x,\hat{\pi}(x)) - \alpha \sum_{y \in \mathbb{X}} p(y \mid x,\hat{\pi}(x)) J_2^*(y) \right|,$$

where we applied that $\forall f_1, f_2 : \mathcal{S} \to \mathbb{R}$ bounded functions such that $\min_s f_1(s) \leq \min_s f_2(s)$ and $\hat{s} = \arg\min_s f_1(s)$, we have $|\min_s f_1(s) - \min_s f_2(s)| \leq |f_1(\hat{s}) - f_2(\hat{s})|$. Then,

$$\forall x \in \mathbb{X} : |J_1^*(x) - J_2^*(x)| \leq |g_1(x,\hat{\pi}(x)) - g_2(x,\hat{\pi}(x))| + \alpha \sum_{y \in \mathbb{X}} p(y \mid x,\hat{\pi}(x)) \, |J_1^*(y) - J_2^*(y)| \leq$$

$$\leq \|g_1 - g_2\|_\infty + \alpha \sum_{y \in \mathbb{X}} p(y \mid x,\hat{\pi}(x)) \, \|J_1^* - J_2^*\|_\infty =$$

$$= \|g_1 - g_2\|_\infty + \alpha \|J_1^* - J_2^*\|_\infty .$$

It is easy to see that if

$$\forall x \in \mathbb{X} : |J_1^*(x) - J_2^*(x)| \leq \|g_1 - g_2\|_\infty + \alpha \|J_1^* - J_2^*\|_\infty ,$$

then

$$\|J_1^* - J_2^*\|_\infty \leq \|g_1 - g_2\|_\infty + \alpha \|J_1^* - J_2^*\|_\infty ,$$

from which the statement of the theorem immediately follows after rearrangement. ■

Theorem 13 *Assume that two discounted MDPs differ only in the discount factors, denoted by* $\alpha_1, \alpha_2 \in [0,1)$. *Let the corresponding optimal value functions be* J_1^* *and* J_2^*, *then*

$$\|J_1^* - J_2^*\|_\infty \leq \frac{|\alpha_1 - \alpha_2|}{(1-\alpha_1)(1-\alpha_2)} \|g\|_\infty .$$

Proof Let π_i^* denote a greedy and deterministic policy based on value function J_i^*, where $i \in \{1,2\}$. Naturally, policy π_i^* is optimal if the discount rate is α_i (Theorem 6). Then, let us introduce a deterministic Markovian control policy $\hat{\pi}$ defined as

$$\hat{\pi}(x) = \begin{cases} \pi_1^*(x) & \text{if } J_1^*(x) \leq J_2^*(x), \\ \\ \pi_2^*(x) & \text{if } J_2^*(x) < J_1^*(x). \end{cases}$$

For any state x the difference of the two value functions can be estimated as follows,

$$|J_1^*(x) - J_2^*(x)| =$$

$$= \left| \min_{a\in\mathcal{A}(x)} \left[g(x,a) + \alpha_1 \sum_{y\in\mathbb{X}} p(y \mid x,a)\, J_1^*(y) \right] - \min_{a\in\mathcal{A}(x)} \left[g(x,a) + \alpha_2 \sum_{y\in\mathbb{X}} p(y \mid x,a)\, J_2^*(y) \right] \right| \leq$$

$$\leq \left| g(x,\hat{\pi}(x)) + \alpha_1 \sum_{y\in\mathbb{X}} p(y \mid x,\hat{\pi}(x))\, J_1^*(y) - g(x,\hat{\pi}(x)) - \alpha_2 \sum_{y\in\mathbb{X}} p(y \mid x,\hat{\pi}(x))\, J_2^*(y) \right|,$$

where we applied that $\forall\, f_1, f_2 : \mathcal{S} \to \mathbb{R}$ bounded functions such that $\min_s f_1(s) \leq \min_s f_2(s)$ and $\hat{s} = \arg\min_s f_1(s)$, we have $|\min_s f_1(s) - \min_s f_2(s)| \leq |f_1(\hat{s}) - f_2(\hat{s})|$. Then,

$$\forall x \in \mathbb{X} : |J_1^*(x) - J_2^*(x)| \leq \left| \sum_{y\in\mathbb{X}} p(y \mid x,\hat{\pi}(x)) \left(\alpha_1 J_1^*(y) - \alpha_2 J_2^*(y)\right) \right| \leq$$

$$\leq |\alpha_1 - \alpha_2| \frac{1}{1-\alpha_1} \|g\|_\infty + \alpha_2 \|J_1^* - J_2^*\|_\infty,$$

where in the last step we used the following estimation of $|\alpha_1 J_1^*(y) - \alpha_2 J_2^*(y)|$,

$$|\alpha_1 J_1^*(y) - \alpha_2 J_2^*(y)| = |\alpha_1 J_1^*(y) - \alpha_2 J_1^*(y) + \alpha_2 J_1^*(y) - \alpha_2 J_2^*(y)| \leq$$

$$\leq |\alpha_1 - \alpha_2|\, |J_1^*(y)| + \alpha_2\, |J_1^*(y) - J_2^*(y)| \leq |\alpha_1 - \alpha_2| \frac{1}{1-\alpha_1} \|g\|_\infty + \alpha_2 \|J_1^* - J_2^*\|_\infty,$$

where we applied the fact that for any state y we have,

$$|J_1^*(y)| \leq \sum_{t=0}^{\infty} \alpha_1^t \|g\|_\infty = \frac{1}{1-\alpha_1} \|g\|_\infty.$$

Because the estimation of $|J_1^*(x) - J_2^*(x)|$ is valid for all x, we have the following result

$$\|J_1^* - J_2^*\|_\infty \leq |\alpha_1 - \alpha_2| \frac{1}{1-\alpha_1} \|g\|_\infty + \alpha_2 \|J_1 - J_2\|_\infty,$$

from which the statement of the theorem immediately follows after rearrangement. ■

Lemma 14 *Assume that we have two discounted MDPs which differ only in the transition-probability functions or only in the immediate-cost functions or only in the discount factors. Let the corresponding optimal action-value functions be Q_1^* and Q_2^*, respectively. Then, the bounds for $\|J_1^* - J_2^*\|_\infty$ of Theorems 11, 12 and 13 are also bounds for $\|Q_1^* - Q_2^*\|_\infty$.*

Proof We will prove the theorem in three parts, depending on the changing components.
Case 1: Assume that the MDPs differ only in the transition functions, denoted by p_1 and p_2. We will prove the same estimation as in the case of Theorem 11, more precisely, that

$$\|Q_1^* - Q_2^*\|_\infty \leq \frac{\alpha \|g\|_\infty}{(1-\alpha)^2} \|p_1 - p_2\|_1.$$

For all state-action pair (x,a) we can estimate the absolute difference of Q_1^* and Q_2^* as

$$|Q_1^*(x,a) - Q_2^*(x,a)| =$$

$$= \left| g(x,a) + \alpha \sum_{y \in \mathbb{X}} p_1(y|x,a) J_1^*(y) - g(x,a) - \alpha \sum_{y \in \mathbb{X}} p_2(y|x,a) J_2^*(y) \right| \leq$$

$$\leq \left| \alpha \sum_{y \in \mathbb{X}} \left(p_1(y|x,a) J_1^*(y) - p_2(y|x,a) J_2^*(y) \right) \right|,$$

from which the proof continues in the same way as the proof of Theorem 11.
Case 2: Assume that the MDPs differ only in the immediate-cost functions, denoted by g_1 and g_2. We will prove the same estimation as in the case of Theorem 12, more precisely,

$$\|Q_1^* - Q_2^*\|_\infty \leq \frac{1}{1-\alpha} \|g_1 - g_2\|_\infty .$$

For all state-action pair (x,a) we can estimate the absolute difference of Q_1^* and Q_2^* as

$$|Q_1^*(x,a) - Q_2^*(x,a)| =$$

$$= \left| g_1(x,a) + \alpha \sum_{y \in \mathbb{X}} p(y|x,a) J_1^*(y) - g_2(x,a) - \alpha \sum_{y \in \mathbb{X}} p(y|x,a) J_2^*(y) \right| \leq$$

$$\leq \|g_1 - g_2\|_\infty + \left| \alpha \sum_{y \in \mathbb{X}} p(y|x,a) (J_1^*(y) - J_2^*(y)) \right| \leq \|g_1 - g_2\|_\infty + \alpha \|J_1^* - J_2^*\|_\infty .$$

The statement immediately follows after we apply Theorem 12 to estimate $\|J_1^* - J_2^*\|_\infty$.
Case 3: Assume that the MDPs differ only in the discount rates, denoted by α_1 and α_2. We will prove the same estimation as in the case of Theorem 13, more precisely, that

$$\|Q_1^* - Q_2^*\|_\infty \leq \frac{|\alpha_1 - \alpha_2|}{(1-\alpha_1)(1-\alpha_2)} \|g\|_\infty .$$

For all state-action pair (x,a) we can estimate the absolute difference of Q_1^* and Q_2^* as

$$|Q_1^*(x,a) - Q_2^*(x,a)| =$$

$$= \left| g(x,a) + \alpha_1 \sum_{y \in \mathbb{X}} p(y|x,a) J_1^*(y) - g(x,a) - \alpha_2 \sum_{y \in \mathbb{X}} p(y|x,a) J_2^*(y) \right| \leq$$

$$\leq \left| \alpha_1 \sum_{y \in \mathbb{X}} p(y|x,a) J_1^*(y) - \alpha_2 \sum_{y \in \mathbb{X}} p(y|x,a) J_2^*(y) \right| \leq |\alpha_1 - \alpha_2| \frac{1}{1-\alpha_1} \|g\|_\infty + \alpha_2 \|J_1^* - J_2^*\|_\infty ,$$

where in the last step we applied the same estimation as in the proof of Theorem 13. The statement immediately follows after we apply Theorem 13 to estimate $\|J_1^* - J_2^*\|_\infty$. ■

Lemma 18 *Assume that two discounted MDPs, $\mathcal{M}_1$ and $\mathcal{M}_2$, differ only in the discount factors, denoted by α_1 and α_2. Then, there exists an MDP, denoted by $\mathcal{M}_3$, such that it differs only in the immediate-cost function from $\mathcal{M}_1$, thus its discount factor is α_1, and it has the same optimal value function as $\mathcal{M}_2$. The immediate-cost function of $\mathcal{M}_3$ is*

$$\widehat{g}(x,a) = g(x,a) + (\alpha_2 - \alpha_1) \sum_{y \in \mathbb{X}} p(y|x,a) J_2^*(y),$$

where p is the probability-transition function of $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$; g is the immediate-cost function of $\mathcal{M}_1$ and $\mathcal{M}_2$; and $J_2^*(y)$ denotes the optimal cost-to-go function of $\mathcal{M}_2$.

Proof First of all, let us overview some general statements that will be used in the proof.

Recall from Bertsekas and Tsitsiklis (1996) that we can treat the solution (the optimal value function) of the infinite horizon problem as the limit of the finite horizon solutions. More precisely, the Bellman optimality equation for the n-stage (finite horizon) problem is

$$J_k^*(x) = \min_{a\in\mathcal{A}(x)} \Big[g(x,a) + \alpha \sum_{y\in\mathbb{X}} p(y \mid x,a)\, J_{k-1}^*(y) \Big],$$

for all $k \in \{1,\ldots,n\}$ and $x \in \mathbb{X}$. Note that by definition, we have $J_0^*(x) = 0$. Moreover,

$$\forall x \in \mathbb{X} : J^*(x) = J_\infty^*(x) = \lim_{n\to\infty} J_n^*(x).$$

Also recall that the n-stage optimal action value function is defined as

$$Q_k^*(x,a) = g(x,a) + \alpha \sum_{y\in\mathbb{X}} p(y|x,a) J_{k-1}^*(y),$$

for all x, a and $k \in \{1,\ldots,n\}$. We also have $Q_0^*(x,a) = 0$ and $J_n^*(x) = \min_a Q_n^*(x,a)$.

During the proof we will apply the solutions of suitable finite horizon problems, thus, in order to avoid notational confusions, let us denote the optimal state and action value functions of $\mathcal{M}_2$ and $\mathcal{M}_3$ by J^*, Q^* and $\hat{J}^*$, $\widehat{Q}^*$, respectively. The corresponding finite horizon optimal value functions will be denoted by J_n^*, Q_n^* and $\hat{J}_n^*$, $\widehat{Q}_n^*$, respectively, where n is the length of the horizon. We will show that for all state x and action a we have $Q^*(x,a) = \widehat{Q}^*(x,a)$, from which $J^* = \hat{J}^*$ follows. Let us define $\widehat{g}_n$ for all $n > 0$ by

$$\widehat{g}_n(x,a) = g(x,a) + (\alpha_2 - \alpha_1) \sum_{y\in\mathbb{X}} p(y|x,a) J_{n-1}^*(y).$$

We will apply induction on n. For the case of $n = 0$ we trivially have $Q_0^* = \widehat{Q}_0^*$, since both of them are constant zero functions. Now, assume that $Q_k^* = \widehat{Q}_k^*$ for $k \le n$, then

$$\widehat{Q}_{n+1}^*(x,a) = \widehat{g}_{n+1}(x,a) + \alpha_1 \sum_{y\in\mathbb{X}} p(y|x,a) \hat{J}_n^*(y) =$$

$$= g(x,a) + (\alpha_2 - \alpha_1) \sum_{y\in\mathbb{X}} p(y|x,a) J_n^*(y) + \alpha_1 \sum_{y\in\mathbb{X}} p(y|x,a) \hat{J}_n^*(y) =$$

$$= g(x,a) + \alpha_2 \sum_{y\in\mathbb{X}} p(y|x,a) J_n^*(y) + \alpha_1 \sum_{y\in\mathbb{X}} p(y|x,a) \big(\hat{J}_n^*(y) - J_n^*(y) \big) =$$

$$= g(x,a) + \alpha_2 \sum_{y\in\mathbb{X}} p(y|x,a) J_n^*(y) + \alpha_1 \sum_{y\in\mathbb{X}} p(y|x,a) \left(\min_{b\in\mathcal{A}(y)} \widehat{Q}_n^*(y,b) - \min_{b\in\mathcal{A}(y)} Q_n^*(y,b) \right) =$$

$$= g(x,a) + \alpha_2 \sum_{y\in\mathbb{X}} p(y|x,a) J_n^*(y) = Q_{n+1}^*(x,a).$$

We have proved that for all n: $Q_n^* = \widehat{Q}_n^*$. Consequently, $Q^*(x,a) = \lim_{n\to\infty} Q_n^*(x,a) = \lim_{n\to\infty} \widehat{Q}_n^*(x,a)$ $= \widehat{Q}^*(x,a)$ and, thus, $J^*(x) = \min_a Q^*(x,a) = \min_a \widehat{Q}^*(x,a) = \hat{J}^*(x)$. Finally, note that for the case

of the infinite horizon problem $\widehat{g}(x,a) = \lim_{n\to\infty} \widehat{g}_n(x,a)$. ■

Theorem 20 *Suppose that Assumptions 1-3 hold and let V_t be the sequence generated by*

$$V_{t+1}(x) = (1-\gamma_t(x))V_t(x) + \gamma_t(x)((K_tV_t)(x) + W_t(x)),$$

then, for any $V^*, V_0 \in \mathcal{V}$, the sequence V_t κ-approximates function V^* with

$$\kappa = \frac{4\rho}{1-\beta_0} \quad \textit{where} \quad \rho = \limsup_{t\to\infty} \|V_t^* - V^*\|_\infty.$$

The applied three main assumptions are as follows
Assumption 1 *There exits a constant $C > 0$ such that for all state x and time t, we have*

$$\mathbb{E}[W_t(x) \,|\, \mathcal{F}_t] = 0 \quad \textit{and} \quad \mathbb{E}\left[W_t^2(x) \,|\, \mathcal{F}_t\right] < C < \infty.$$

Assumption 2 *For all x and t, $0 \le \gamma_t(x) \le 1$, and we have with probability one*

$$\sum_{t=0}^{\infty} \gamma_t(x) = \infty \quad \textit{and} \quad \sum_{t=0}^{\infty} \gamma_t^2(x) < \infty.$$

Assumption 3 *For all t, operator $K_t : \mathcal{V} \to \mathcal{V}$ is a supremum norm contraction mapping with Lipschitz constant $\beta_t < 1$ and with fixed point V_t^*. Formally, for all $V_1, V_2 \in \mathcal{V}$,*

$$\|K_tV_1 - K_tV_2\|_\infty \le \beta_t \|V_1 - V_2\|_\infty.$$

Let us introduce a common Lipschitz constant $\beta_0 = \limsup_{t\to\infty} \beta_t$, and assume that $\beta_0 < 1$.

Proof During the proof, our main aim will be to apply Theorem 9, thus, we have to show that the assumptions of the theorem hold. Let us define operator H_t for all $V_a, V_b \in \mathcal{V}$ by

$$H_t(V_a, V_b)(x) = (1-\gamma_t(x))V_a(x) + \gamma_t(x)((K_tV_b)(x) + W_t(x)).$$

Applying this definition, first, we will show that $V'_{t+1} = H_t(V'_t, V^*)$ κ-approximates V^* for all V'_0. Because $\beta_t < 1$ for all t and $\limsup_{t\to\infty} \beta_t = \beta_0 < 1$, it follows that $\sup_t \beta_t = \widetilde{\beta} < 1$ and each K_t is $\widetilde{\beta}$ contraction. We know that $\limsup_{t\to\infty} \|V^* - V_t^*\|_\infty = \rho$, therefore, for all $\delta > 0$, there is an index t_0 such that for all $t \ge t_0$, we have that $\|V^* - V_t^*\|_\infty \le \rho + \delta$. Using these observations, we can estimate $\|K_tV^*\|_\infty$ for all $t > t_0$, as follows

$$\|K_tV^*\|_\infty = \|K_tV^* - V^* + V^*\|_\infty \le \|K_tV^* - V^*\|_\infty + \|V^*\|_\infty \le$$

$$\le \|K_tV^* - V_t^* + V_t^* - V^*\|_\infty + \|V^*\|_\infty \le \|K_tV^* - V_t^*\|_\infty + \|V_t^* - V^*\|_\infty + \|V^*\|_\infty \le$$

$$\le \|K_tV^* - K_tV_t^*\|_\infty + \rho + \delta + \|V^*\|_\infty \le \widetilde{\beta}\|V^* - V_t^*\|_\infty + \rho + \delta + \|V^*\|_\infty \le$$

$$\le (1+\widetilde{\beta})\rho + (1+\widetilde{\beta})\delta + \|V^*\|_\infty \le (1+\widetilde{\beta})\rho + 2\delta + \|V^*\|_\infty.$$

If we apply $\delta = (1-\widetilde{\beta})\rho/2$, then for sufficiently large t $(t \ge t_0)$ we have that

$$\|K_tV^*\|_\infty \le 2\rho + \|V^*\|_\infty.$$

Now, we can upper estimate $V'_{t+1} = H_t(V'_t, V^*)$, for all $x \in \mathcal{X}$, $V'_0 \in \mathcal{V}$ and $t \geq t_0$ by

$$V'_{t+1}(x) = H_t(V'_t, V^*)(x) = (1-\gamma_t(x))V'_t(x) + \gamma_t(x)((K_t V^*)(x) + W_t(x)) \leq$$

$$\leq (1-\gamma_t(x))V'_t(x) + \gamma_t(x)(\|K_t V^*\|_\infty + W_t(x)) \leq$$

$$\leq (1-\gamma_t(x))V'_t(x) + \gamma_t(x)(\|V^*\|_\infty + 2\rho + W_t(x)).$$

Let us define a new sequence for all $x \in \mathcal{X}$ by

$$\widetilde{V}_{t+1}(x) = \begin{cases} (1-\gamma_t(x))\widetilde{V}_t(x) + \gamma_t(x)(\|V^*\|_\infty + 2\rho + W_t(x)) & \text{if } t \geq t_0, \\ V'_t(x) & \text{if } t < t_0. \end{cases}$$

It is easy to see (for example, by induction from t_0) that for all time t and state x we have that $V'_t(x) \leq \widetilde{V}_t(x)$ with probability one, more precisely, for almost all $\omega \in \Omega$, where $\omega = \langle \omega_1, \omega_2, \dots \rangle$ drives the noise parameter $W_t(x) = w_t(x, \omega_t)$ in both V'_t and $\widetilde{V}_t$. Note that Ω is the sample space of the underlying probability measure space $\langle \Omega, \mathcal{F}, \mathbb{P} \rangle$.

Applying the "Conditional Averaging Lemma" of Szepesvári and Littman (1999), which is a variant of the Robbins-Monro Theorem and requires Assumptions 1 and 2, we get that $\widetilde{V}_t(x)$ converges with probability one to $2\rho + \|V^*\|_\infty$ for all $\widetilde{V}_0 \in \mathcal{V}$ and $x \in \mathcal{X}$. Therefore, because $V'_t(x) \leq \widetilde{V}_t(x)$ for all x and t with probability one, we have that the sequence $V'_t(x)$ κ-approximates $V^*(x)$ with $\kappa = 2\rho$ for all function $V'_0 \in \mathcal{V}$ and state $x \in \mathcal{X}$.

Now, let us turn to Conditions 1-4 of Theorem 9. For all x and t, we define functions $F_t(x)$ and $G_t(x)$ as $F_t(x) = \beta_t \gamma_t(x)$ and $G_t(x) = (1-\gamma_t(x))$. By Assumption 2, functions $F_t(x), G_t(x) \in [0,1]$ for all x and t. Condition 1 trivially follows from the definitions of G_t and H_t. For the proof of Condition 2, we need Assumption 3, namely that each operator K_t is a contraction with respect to β_t. Condition 3 is a consequence of Assumption 2 and the definition of G_t. Finally, we have Condition 4 for any $\varepsilon > 0$ and sufficiently large t by defining $\xi = \beta_0 + \varepsilon$. Applying Theorem 9 with $\kappa = 2\rho$, we get that V_t κ'-approximates V^* with $\kappa' = 4\rho/(1-\beta_0-\varepsilon)$. In the end, letting ε go to zero proves our statement. ■

References

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, Massachusetts, 3rd edition, 2007.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

B. Cs. Csáji. *Adaptive Resource Control: Machine Learning Approaches to Resource Allocation in Uncertain and Changing Environments*. PhD thesis, Faculty of Informatics, Eötvös Loránd University, Budapest, 2008.

B. Cs. Csáji and L. Monostori. Adaptive sampling based large-scale stochastic resource control. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06), July 16-20, Boston, Massachusetts*, pages 815–820, 2006.

R. Montes de Oca, A. Sakhanenko, and F. Salem. Estimates for perturbations of general discounted Markov control chains. *Applied Mathematics*, 30:287–304, 2003.

E. Even-Dar and Y. Mansour. Learning rates for Q-learning. *Journal of Machine Learning Research (JMLR)*, 5:1–25, Dec. 2003.

G. Favero and W. J. Runggaldier. A robustness result for stochastic control. *Systems and Control Letters*, 46:91–66, 2002.

E. A. Feinberg and A. Shwartz, editors. *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer Academic Publishers, 2002.

E. Gordienko and F. S. Salem. Estimates of stability of Markov control processes with unbounded cost. *Kybernetika*, 36:195–210, 2000.

Zs. Kalmár, Cs. Szepesvári, and A. Lőrincz. Module-based reinforcement learning: Experiments with a real robot. *Machine Learning*, 31:55–85, 1998.

M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

A. Müller. How does the solution of a Markov decision process depend on the transition probabilities? Technical report, Institute for Economic Theory and Operations Research, University of Karlsruhe, 1996.

R. Munos and A. W. Moore. Rates of convergence for variable resolution schemes in optimal control. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 647–654. Morgan Kaufmann, San Francisco, CA, 2000.

M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, 2002.

S. Singh and D. Bertsekas. Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems*, volume 9, pages 974–980. The MIT Press, 1997.

R. S. Sutton and A. G. Barto. *Reinforcement Learning*. The MIT Press, 1998.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12: 1057–1063, 2000.

Cs. Szepesvári and M. L. Littman. A unified analysis of value-function-based reinforcement learning algorithms. *Neural Computation*, 11(8):2017–2060, 1999.

I. Szita, B. Takács, and A. Lőrincz. ε-MDPs: Learning in varying environments. *Journal of Machine Learning Research (JMLR)*, 3:145–174, 2002.

B. Van Roy, D. Bertsekas, Y. Lee, and J. Tsitsiklis. A neuro-dynamic programming approach to retailer inventory management. Technical report, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA., 1996.

Journal of Machine Learning Research 9 (2008) 1711-1739 Submitted 8/06; Revised 3/07; Published 8/08

Regularization on Graphs with Function-adapted Diffusion Processes

Arthur D. Szlam ASZLAM@MATH.UCLA.EDU
Department of Mathematics
U.C.L.A., Box 951555
Los Angeles, CA 90095-1555

Mauro Maggioni MAURO.MAGGIONI@DUKE.EDU
Department of Mathematics and Computer Science
Duke University, Box 90320
Durham, NC, 27708

Ronald R. Coifman COIFMAN@MATH.YALE.EDU
Program in Applied Mathematics
Department of Mathematics
Yale University, Box 208283
New Haven,CT,06510

Editor: Zoubin Ghahrmani

Abstract

Harmonic analysis and diffusion on discrete data has been shown to lead to state-of-the-art algorithms for machine learning tasks, especially in the context of semi-supervised and transductive learning. The success of these algorithms rests on the assumption that the function(s) to be studied (learned, interpolated, etc.) are smooth with respect to the geometry of the data. In this paper we present a method for modifying the given geometry so the function(s) to be studied are smoother with respect to the modified geometry, and thus more amenable to treatment using harmonic analysis methods. Among the many possible applications, we consider the problems of image denoising and transductive classification. In both settings, our approach improves on standard diffusion based methods.

Keywords: diffusion processes, diffusion geometry, spectral graph theory, image denoising, transductive learning, semi-supervised learning

1. Introduction

Recently developed techniques in the analysis of data sets and machine learning use the geometry of the data set in order to study functions on it. In particular the idea of analyzing the data set and functions on it intrinsically has lead to novel algorithms with state-of-the-art performance in various problems in machine learning (Szummer and Jaakkola, 2001; Zhu et al., 2003; Zhou and Schlkopf, 2005; Belkin and Niyogi, 2003a; Mahadevan and Maggioni, 2007; Maggioni and Mhaskar, 2007). They are based on the construction of a diffusion, or an averaging operator K on the data set, dependent on its local, fine scale geometry. K, its powers, and the special bases associated to it, such as its eigenfunctions (Belkin and Niyogi, 2003a; Coifman et al., 2005a; Coifman and Lafon, 2006a) or its diffusion wavelets (Coifman and Maggioni, 2006) can be used to study the geometry of and analyze functions on the data set. Among other things, "diffusion analysis" allows us to introduce a notion of smoothness in discrete settings that preserves the relationships between smoothness,

sparsity in a "Fourier" basis, and evolution of heat that are well-known in Euclidean spaces (Zhou and Schlkopf, 2005).

One of the main contributions of this work is the observation that the geometry of the space is not the only important factor to be considered, but that the geometry and the properties of the function f to be studied (denoised/learned) should also affect the smoothing operation of the diffusion. We will therefore modify the geometry of a data set with features from f, and build K on the modified f-adapted data set. The reason for doing this is that perhaps f is not smooth with respect to the geometry of the space, but has structure that is well encoded in the features. Since the harmonic analysis tools we use are robust to complicated geometries, but are most useful on smooth functions, it is reasonable to let the geometry of the data set borrow some of the complexity of the function, and study a smoother function on a more irregular data set. In other words, we attempt to find the geometry so that the functions to be studied are as smooth as possible with respect to that geometry. On the one hand, the result is nonlinear in the sense that it depends on the input function f, in contrast with methods which consider the geometry of the data alone, independently of f. On the other hand, on the modified data set, the smoothing operator K will be linear, and very efficiently computable. One could generalize the constructions proposed to various types of processes (e.g., nonlinear diffusions).

The paper is organized as follows: in Section 2, we review the basic ideas of harmonic analysis on weighted graphs. In Section 3 we introduce the function-adapted diffusion approach, which aims to modify the geometry of a data set so that a function or class of functions which was non-smooth in the original geometry is smooth in the modified geometry, and thus amenable to the harmonic analysis in the new geometry. In Section 4 we demonstrate this approach in the context of the image denoising problem. In addition to giving easy to visualize examples of how the method works, we achieve state of the art results. In Section 5, we demonstrate the approach in the context of transductive learning. While here it is more difficult to interpret our method visually, we test it on a standard database, where it outperforms comparable "geometry only" methods on the majority of the data sets, and in many cases achieves state of the art results. We conclude by considering the under-performance of the method on some data sets, observing that in those examples (most of which are in fact the only artificial ones!), the geometry of the data suffices for learning the function of interests; and our method is superfluous.

2. Diffusion on Graphs Associated with Data-sets

An intrinsic analysis of a data set, modeled as a graph or a manifold, can be developed by considering a natural random walk K on it (Chung, 1997; Szummer and Jaakkola, 2001; Ng et al., 2001; Belkin and Niyogi, 2001; Zha et al., 2001; Lafon, 2004; Coifman et al., 2005a,b). The random walk allows to construct diffusion operators on the data set, as well as associated basis functions. For an initial condition δ_x, $K^t\delta_x(y)$ represents the probability of being at y at time t, conditioned on starting at x.

2.1 Setup and Notation

We consider the following general situation: the space is a finite weighted graph $G = (V,E,W)$, consisting of a set V (vertices), a subset E (edges) of $V \times V$, and a nonnegative function $W : E \to \mathbb{R}^+$ (weights). Without loss of generality we assume that there is an edge from x to $y \in V$, and write $x \sim y$, if and only if $W(x,y) > 0$. Notice that in this work W will usually be symmetric; that is

the edges will be undirected. The techniques we propose however do not require this property, and therefore can be used on data sets in which graph models with directed edges are natural.

We interpret the weight $W(x,y)$ as a measure of similarity between the vertices x and y. A natural filter acting on functions on V can be defined by normalization of the weight matrix as follows: let

$$d(x) = \sum_{y \in V} W(x,y)\,,$$

and let[1] the filter be

$$K(x,y) = d^{-1}(x)W(x,y)\,, \tag{1}$$

so that $\sum_{y\in V} K(x,y) = 1$, and so that multiplication Kf of a vector from the left is a local averaging operation, with locality measured by the similarities W. Multiplication by K can also be interpreted as a generalization of Parzen window type estimators to functions on graphs/manifolds. There are other ways of defining averaging operators. For example one could consider the heat kernel $e^{-t\mathcal{L}}$ where $\mathcal{L}$ is defined in (3) below, see also Chung (1997), or a bi-Markov matrix similar to W (Sinkhorn, 1964; Sinkhorn and Knopp, 1967; Soules, 1991; Linial et al., 1998; Shashua et al., 2005; Zass and Shashua, 2005).

In general K is not column-stochastic,[2] but the operation fK of multiplication on the right by a (row) vector can be thought of as a diffusion of the vector f. This filter can be iterated several times by considering the power K^t.

2.2 Graphs Associated with Data Sets

¿From a data set X we construct a graph G: the vertices of G are the data points in X, and weighted edges are constructed that connect nearby data points, with a weight that measures the similarity between data points. The first step is therefore defining these *local similarities*. This is a step which is data- and application-dependent. It is important to stress the attribute *local*. Similarities between far away data points are not required, and deemed unreliable, since they would not take into account the geometric structure of the data set. Local similarities are assumed to be more reliable, and non-local similarities will be inferred from local similarities through diffusion processes on the graph.

2.2.1 LOCAL SIMILARITIES

Local similarities are collected in a matrix W, whose rows and columns are indexed by X, and whose entry $W(x,y)$ is the similarity between x and y. In the examples we consider here, W will usually be symmetric, that is the edges will be undirected, but these assumptions are not necessary.

If the data set lies in $\mathbb{R}^d$, or in any other metric space with metric ρ, then the most standard construction is to choose a number ("local time") $\sigma > 0$ and let

$$W_\sigma(x,y) = h\left(\frac{\rho(x,y)^2}{\sigma}\right),$$

1. Note that $d(x) = 0$ if and only if x is not connected to any other vertex, in which case we trivially define $d^{-1}(x) = 0$, or simply remove x from the graph.
2. In particular cases K is a scalar multiple of a column-stochastic matrix, for example when D is a multiple of identity, which happens for example if G is regular and all the edges have the same weight.

for some function h with, say, exponential decay at infinity. A common choice is $h(a) = \exp(-a)$. The idea is that we expect that very close data points (with respect to ρ) will be similar, but do not want to assume that far away data points are necessarily different.

Let D be the diagonal matrix with entries given by d as in (2.1). Suppose the data set is, or lies on, a manifold in Euclidean space. In Lafon (2004), Belkin and Niyogi (2005), Hein et al. (2005), von Luxburg et al. (2004) and Singer (2006), it is proved that in this case, the choice of h in the construction of the weight matrix is in some asymptotic sense irrelevant. For a rather generic symmetric function h, say with exponential decay at infinity, $(I - D_\sigma^{-\frac{1}{2}} W_\sigma D_\sigma^{-\frac{1}{2}})/\sigma$, approaches the Laplacian on the manifold, at least in a weak sense, as the number of points goes to infinity and σ goes to zero. Thus this choice of weights is naturally related to the heat equation on the manifold. On the other hand, for many data sets, which either are far from asymptopia or simply do not lie on a manifold, the choice of weights can make a large difference and is not always easy. Even if we use Gaussian weights, the choice of the "local time parameter" σ can be nontrivial.

For each x, one usually limits the maximum number of points y such that $W(x,y) \neq 0$ (or non-negligible). Two common modifications of the construction above are to use either $\rho_\varepsilon(x,y)$ or $\rho_k(x,y)$ instead of ρ, where

$$\rho_\varepsilon(x,y) = \begin{cases} d(x,y) & \text{if } \rho(x,y) \leq \varepsilon; \\ \infty & \text{if } \rho(x,y) > \varepsilon \end{cases},$$

where usually ε is such that $h(\varepsilon^2/\sigma) << 1$, and

$$\rho_k(x,y) = \begin{cases} \rho(x,y) & \text{if } y \in n_k(x); \\ \infty & \text{otherwise.} \end{cases}$$

and $n_k(x)$ is the set of k nearest neighbors of x. This is for two reasons: one, often only very small distances give information about the data points, and two, it is usually only possible to work with very sparse matrices.[3] This truncation causes W to be not symmetric; if symmetry is desired, W may be averaged (arithmetically or geometrically) with its transpose.

A location-dependent approach for selecting the similarity measure is suggested in Zelnik-Manor and Perona (2004). A number m is fixed, and the distances at each point are scaled so the m-th nearest neighbor has distance 1; that is, we let $\rho_x(y,y') = \rho(y,y')/\rho(x,x_m)$, where x_m is the m-th nearest neighbor to x. Now ρ_x depends on x, so in order to make the weight matrix symmetric, they suggest to use the geometric mean of ρ_x and ρ_y in the argument of the exponential, that is, let

$$W_\sigma(x,y) = h\left(\frac{\rho_x(x,y)\rho_y(x,y)}{\sigma}\right), \tag{2}$$

with h, as above, decaying at infinity (typically, $h(a) = \exp(-a)$), or truncated at the k-th nearest neighbor. This is called the self-tuning weight matrix. There is still a timescale in the weights, but a global σ in the self-tuning weights corresponds to some location dependent choice of σ in the standard exponential weights.

3. However, methods of Fast Multipole of Fast Gauss type (Greengard and Rokhlin, 1988) may make it possible to work with dense matrices implicitly, with complexity proportional to the number of points. See Raykar et al. (2005) for a recent reference with applications to machine learning.

2.2.2 THE AVERAGING OPERATOR AND ITS POWERS

Multiplication by the normalized matrix K as in (1) can be iterated to generate a Markov process $\{K^t\}_{t\geq 0}$, and can be used to measure the strength of all the paths between two data points, or the likelihood of getting from one data point to the other if we constrain ourselves to only stepping between very similar data points. For example one defines the diffusion or spectral distance (Bérard et al., 1994; Coifman et al., 2005a; Coifman and Lafon, 2006a) by

$$\mathcal{D}^{(t)}(x,y) = ||K^t(x,\cdot) - K^t(y,\cdot)||_2 = \sqrt{\sum_{z\in X} |K^t(x,z) - K^t(y,z)|^2}\,.$$

The term diffusion distance was introduced in Lafon (2004), Coifman et al. (2005a) and Coifman and Lafon (2006a) and is suggested by the formula above, which expresses $\mathcal{D}^{(t)}$ as some similarity between the probability distributions $K^t(x,\cdot)$ and $K^t(y,\cdot)$, which are obtained by diffusion from x and y according to the diffusion process K. The term spectral distance was introduced in Bérard et al. (1994, see also references therein). It has recently inspired several algorithms in clustering, classification and learning (Belkin and Niyogi, 2003a, 2004; Lafon, 2004; Coifman et al., 2005a; Coifman and Lafon, 2006a; Mahadevan and Maggioni, 2005; Lafon and Lee, to appear, 2006; Maggioni and Mhaskar, 2007).

2.3 Harmonic Analysis

The eigenfunctions $\{\psi_i\}$ of K, satisfying

$$K\psi_i = \lambda_i \psi_i\,,$$

are are related,via multiplication by $D^{-\frac{1}{2}}$, to the eigenfunctions ϕ_i of the graph Laplacian (Chung, 1997), since

$$\mathcal{L} = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} - I = D^{\frac{1}{2}} K D^{-\frac{1}{2}} - I\,. \tag{3}$$

They lead to a natural generalization of the Fourier analysis: any function $g \in \mathbb{L}^2(X)$ can be written as $g = \sum_{i\in I} \langle g, \phi_i\rangle \phi_i$, since $\{\phi_i\}$ is an orthonormal basis. The larger is i, the more oscillating the function ϕ_i is, with respect to the geometry given by W, and λ_i measures the frequency of ϕ_i. These eigenfunctions can be used for dimensionality reduction tasks (Lafon, 2004; Belkin and Niyogi, 2003a; Coifman and Lafon, 2006a; Coifman et al., 2005a; Jones et al., 2007a,b).

For a function g on G, define its gradient (Chung, 1997; Zhou and Schlkopf, 2005) as the function on the edges of G defined by

$$\nabla g(x,y) = W(x,y)\left(\frac{g(y)}{\sqrt{d(y)}} - \frac{g(x)}{\sqrt{d(x)}}\right) \tag{4}$$

if there is an edge e connecting x to y and 0 otherwise; then

$$||\nabla g(x)||^2 = \sum_{x\sim y} |\nabla g(x,y)|^2.$$

The smoothness of g can be measured by the Sobolev norm

$$||g||^2_{\mathcal{H}^1} = \sum_x |g(x)|^2 + \sum_x ||\nabla g(x)||^2\,. \tag{5}$$

The first term in this norm measures the size of the function g, and the second term measures the size of the gradient. The smaller $||g||_{\mathcal{H}^1}$, the smoother is g. Just as in the Euclidean case,

$$||g||^2_{\mathcal{H}^1} = ||g|^2_{\mathbb{L}^2(X,d)} - \langle g, \mathcal{L}g \rangle ;$$

thus projecting a function onto the first few terms of its expansion in the eigenfunctions of $\mathcal{L}$ is a smoothing operation.[4]

We see that the relationships between smoothness and frequency forming the core ideas of Euclidean harmonic analysis are remarkably resilient, persisting in very general geometries. These ideas have been applied to a wide range of tasks in the design of computer networks, in parallel computation, clustering (Ng et al., 2001; Belkin and Niyogi, 2001; Zelnik-Manor and Perona, 2004; Kannan et al., 2004; Coifman and Maggioni, 2007), manifold learning (Bérard et al., 1994; Belkin and Niyogi, 2001; Lafon, 2004; Coifman et al., 2005a; Coifman and Lafon, 2006a), image segmentation (Shi and Malik, 2000), classification (Coifman and Maggioni, 2007), regression and function approximation (Belkin and Niyogi, 2004; Mahadevan and Maggioni, 2005; Mahadevan et al., 2006; Mahadevan and Maggioni, 2007; Coifman and Maggioni, 2007).

2.4 Regularization by Diffusion

It is often useful to find the smoothest function $\tilde{f}$ on a data set X with geometry given by W, so that for a given f, $\tilde{f}$ is not too far from f; this task is encountered in problems of denoising and function extension. In the denoising problem, we are given a function $f + \eta$ from $X \to \mathbb{R}$, and η is Gaussian white noise of a given variance, or if one is ambitious, some other possibly stochastic contamination. We must find f. In the function extension or interpolation problem, a relatively large data set is given, but the values of f are known at only relatively few "labeled" points, and the task is to find f on the "unlabeled" points. Both tasks, without any a priori information on f, are impossible; the problems are underdetermined. On the other hand, it is often reasonable to assume f should be smooth, and so we are led to the problem of finding a smooth $\tilde{f}$ close to f.

In Euclidean space, a classical method of mollification is to run the heat equation for a short time with initial condition specified by f. It turns out that the heat equation makes perfect sense on a weighted graph: if f is a function on V, set $f_0 = f$, and $f_{k+1} = Kf$. If $g_k(x) = d^{\frac{1}{2}}(x) f_k(x)$,

$$g_{k+1} - g_k = \mathcal{L} g_k ,$$

so multiplication by K is a step in the evolution of the (density normalized) heat equation. Furthermore, a quick calculation shows this is the gradient descent for the smoothness energy functional $\sum ||\nabla g||^2$. We can thus do "harmonic" interpolation on X by iterating K (Zhu et al., 2003).

We can design more general mollifiers using an expansion on the eigenfunctions $\{\psi_i\}$ of K. For the rest of this section, suppose all inner products are taken against the measure d, that is, $\langle a, b \rangle = \sum a(x)b(x)d(x)$, and so ψ are orthonormal. Then $f = \sum \langle f, \psi_i \rangle \psi_i$ and one can define $\tilde{f}$, a smoothed version of f, by

$$\tilde{f} = \sum_i \alpha_i \langle f, \psi_i \rangle \psi_i$$

4. However it is well known that if f does not have uniform smoothness everywhere, the approximation by eigenfunctions is poor not only in regions of lesser smoothness, but the poor approximation spills to regions of smoothness as well. This lack of localization can be avoided with the multiscale constructions in Coifman and Maggioni (2006) and Maggioni and Mhaskar (2007).

for some sequence $\{\alpha_i\}$ which tends to 0 as $i \to +\infty$; in the interpolation problem, we can attempt to estimate the inner products $\langle f, \psi_i \rangle$, perhaps by least squares. Typical examples for α_i are:

(i) $\alpha_i = 1$ if $i < I$, and 0 otherwise (pure low-pass filter); I usually depends on a priori information on η, for example on the variance of η. This is a band-limited projection (with band I), see for example Belkin (2003).

(ii) $\alpha_i = \lambda_i^t$ for some $t > 0$, this corresponds to setting $\tilde{f} = K^t(f)$, that is, kernel smoothing on the data set, with a data-dependent kernel (Smola and Kondor, 2003; Zhou and Schlkopf, 2005; Chapelle et al., 2006).

(iii) $\alpha_i = P(\lambda_i)$, for some polynomial (or rational function) P, generalizing (ii). See, for example, Maggioni and Mhaskar (2007)

As mentioned, one can interpret $K^t f$ as evolving a heat equation on the graph with an initial condition specified by f. If we would like to balance smoothing by K with fidelity to the original f, we can choose $\beta > 0$ and set $f_0 = f$ and $f_{t+1} = (K f_t + \beta f)/(1+\beta)$; the original function is treated as a heat source. This corresponds at equilibrium to

(iv) $\alpha_i = \beta/(1+\beta-\lambda_i)$.

One can also consider families of nonlinear mollifiers, of the form

$$\tilde{f} = \sum_i m(\langle f, \psi_i \rangle)\psi_i \,,$$

where for example m is a (soft-)thresholding function (Donoho and Johnstone, 1994). In fact, m may be made even dependent on i. While these techniques are classical and well-understood in Euclidean space (mostly in view of applications to signal processing), it is only recently that research in their application to the analysis of functions on data sets has begun (in view of applications to learning tasks, see in particular Maggioni and Mhaskar 2007).

All of these techniques clearly have a regularization effect. This can be easily measured in terms of the Sobolev norm defined in (5): the methods above correspond to removing or damping the components of f (or $f+\eta$) in the subspace spanned by high-frequency ψ_i, which are the ones with larger Sobolev norm.

3. Function-adapted Kernels

The methods above are based on the idea that the function f to be recovered should be smooth with respect to W, but it can happen that an interesting function on data is not smooth with respect to the given geometry on that data. In this case we cannot directly bring to bear the full power of the methods described above. On the other hand, we have seen that these methods are well defined on any weighted graph. We thus propose to modify the geometry of W so that the function(s) to be recovered are as smooth as possible in the modified geometry. Even when f is not smooth, the geometry of W and f can interact in a structured way. We will attempt to incorporate the geometry of the function f (or a family of functions F) in the construction of the weights W; the hope is that we can convert structure to smoothness, and apply the methods of harmonic analysis to a smoother function on a rougher data set. In other words, we want our regularizer to take averages between

points where f has similar structure, in addition to being near to each other in terms of the given geometry of the data.

The simplest version of this idea is to only choose nonzero weights between points on the same level set of f. Then $||\nabla f||$ (with respect to W) is zero everywhere, and the function to be recovered is as smooth as possible. Of course knowing the level sets of f is far too much to ask for in practice. For example, in the function extension problem, if f has only a few values (e.g., for a classification task), knowing the level sets of f would be equivalent to solving the problem.

If we had some estimate $\tilde{f}$ for f, we could set

$$W^f(x,y) = \exp\left(-\frac{||x-y||^2}{\sigma_1} - \frac{|\tilde{f}(x)-\tilde{f}(y)|^2}{\sigma_2}\right), \tag{6}$$

so that when $\sigma_2 << \sigma_1$, the associated averaging kernel K will average locally, but much more along the (estimated) level sets of f than across them, because points on different level sets now have very weak or no affinity. This is related to ideas in Yaroslavsky (1985); Smith and Brady (1995) and Coifman et al. (2005a).

The estimate $\tilde{f}$ of f is just a simple example of a feature map. More generally, we set

$$W^f(x,y) = h_1\left(-\frac{\rho_1(x,y)^2}{\sigma_1}\right) h_2\left(-\frac{\rho_2(\mathcal{F}(f)(x),\mathcal{F}(f)(y))^2}{\sigma_2}\right), \tag{7}$$

where $\mathcal{F}(f)(x)$ is a set of features associated with f, evaluated at the data point x, ρ_1 is a metric on the data set, ρ_2 is a metric on the set of features, h_1 and h_2 are (usually exponentially) decaying functions, and σ_1 and σ_2 are "local time" parameters in data and feature space respectively. Such a similarity is usually further restricted as described at the end of Section 2.2.1. The idea here is to be less ambitious than (6), and posit affinity between points where we strongly believe f to have the same structure, and not necessarily between every point on an (estimated) level set. The averaging matrix K^f associated with W^f can then be used for regularizing, denoising and learning tasks, as described above. We call such a kernel a function-adapted kernel.

The way the function f affects the construction of K^f will be application- and data- specific, as we shall see in the applications to image denoising and graph transductive learning. For example, in the application to image denoising, $\mathcal{F}(f)(x)$ may be a vector of filter responses applied to the image f at location x. In the application to transductive classification, we are given C functions χ_i, defined by $\chi_i(x) = 1$ if x is labeled as a point in class i, and 0 otherwise (either the point is not labeled, or it is not in class i). We set $f = (\chi_i)_{i=1}^N$. Then $\mathcal{F}(f)(x)$ can be obtained by evaluating $K^t(\chi_i)$ at x, where K is a diffusion operator which only depends on the data set, and not on the χ_i's. In all applications, our idea is simply to to try to choose similarities, with the limited information about the function(s) to be recovered that we are given, so that the function(s) are as regular as possible with respect to the chosen similarities.

4. Application I: Denoising of Images

We apply function-adapted kernels to the task of denoising images. Not only this will be helpful to gain intuition about the ideas in Section 3 in a setting where our methods are easily visualized, but it also leads to state-of-art results.

Gray-scale images are often modeled as real-valued functions, or distributions, on Q, a fine discretization of the square $[0,1]^2$, and they are often analyzed, denoised, compressed, inpainted, deblurred as such, see for example Tschumperle (2002), Perona and Malik (1990), Rudin et al. (1992),

Chan and Shen (2005), Perona and Malik (1990), Tomasi and Manduchi (1998), Elad (2002), Boult et al. (1993), Chin and Yeh (1983), Davis and Rosenfeld (1978), Graham (1961), Huang et al. (1979), Lee (1980), Yin et al. (1996) and references therein. It is well known that images are not smooth as functions from Q to $\mathbb{R}$, and in fact the interesting and most important features are often exactly the non-smooth parts of f. Thus Fourier analysis and the heat equation on Q are not ideally suited for images; much of the work referenced above aims to find partial differential equations whose evolution smooths images without blurring edges and textures.

With the approach described in Section 3, unlike with many PDE-based image processing methods, the machinery of smoothing is divorced from the task of feature extraction. We build a graph $G(I)$ whose vertices are the pixels of the image and whose weights are adapted to the image structure, and use the diffusion on the graph with a fidelity term, as described in Section 2.4 to smooth the image, *considered as a function on the graph*. If we are able to encode image structure in the geometry of the graph in such a way that the image is actually smooth as a function on its graph, then the harmonic analysis on the graph will be well-suited for denoising that image. Of course, we have shifted part of the problem to feature extraction, but we will see that very simple and intuitive techniques produce state of the art results.

4.1 Image-adapted Graphs and Diffusion Kernels

To build the image-adapted graph we first associate a feature vector to each location x in the image I, defined on a square Q. A simple choice of $d+2$ dimensional feature vectors is obtained by setting two of the coordinates of the feature vector to be scaled versions of the coordinates of the corresponding pixel in the image αx, where $\alpha \geq 0$ is a scalar, and $x \in Q$. The remaining d features are the responses to convolution with d different filters $g_1, \cdots, g_d$, evaluated at location x. More formally, we pick a d-vector $g = (g_1, \cdots, g_d)$ of filters (i.e., real valued functions on Q), fix $\alpha \geq 0$, and map Q into $\mathbb{R}^{d+2}$ by a *feature map*

$$\begin{aligned} \mathcal{F}_{g,\alpha}(I) : Q &\to \mathbb{R}^{d+2} \\ x &\mapsto (\alpha x, f * g_1(x), \cdots, f * g_d(x)) \end{aligned}$$

This is an extremely flexible construction, and there are many interesting choices for the filters $\{g_i\}$. One could take a few wavelets or curvelets at different scales, or edge filters, or patches of texture, or some measure of local statistics. Also note there are many other choices of feature maps that are not obtained by convolution, see Section 4.1.2 for examples.

The graph $G(I)$ will have vertices given by $\mathcal{F}_{g,\alpha}(x)$, $x \in Q$. To obtain the weighted edges, set

$$\rho(x,y) = \rho_{g,\alpha}(x,y) = ||\mathcal{F}_{g,\alpha}(f)(x) - \mathcal{F}_{g,\alpha}(f)(y)||,$$

where $||\cdot||$ is a norm (e.g., Euclidean) in $\mathbb{R}^{d+2}$. The parameter α specifies the amount of weight to give to the original 2-d space coordinates of the pixels, and may be 0. Alternatively, instead of using a weight α, one can choose sets $S = S(x) \subset Q$ so that

$$\rho(x,y) = d_{g,S}(x,y) = \begin{cases} \rho_{g,0}(x,y) & \text{if } y \in S(x); \\ \infty & \text{otherwise.} \end{cases} . \tag{8}$$

In the discrete case, if we choose $S(x)$ to be the n nearest neighbors of x in the 2 space coordinates: we will write $\rho_{g,n}$, and if the filters are understood, just ρ_n.

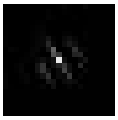


Figure 1: Above: image of Lena, with two locations highlighted. Left: row of the diffusion kernel corresponding to the upper-left highlighted area in the above image. Right: row of the diffusion kernel corresponding to the bottom-left highlighted area in the above image. The diffusion kernel averages according to different patterns in different locations. The averaging pattern on the right is also "non-local", in the sense that the averaging occurs along well-separated stripes, corresponding to the hair in the original picture.

For a fixed choice of metric ρ as above, and a "local time" parameter σ, we construct the similarity matrix W_σ as described in Section 2.2.1, and the associated diffusion kernel K as in (1).

In Figure 3 we explore the local geometry in patch space by projecting the set of patches around a given patch onto the principal components of the set of patches itself. Geometric structures of the set of patches, dependent on local geometry of the image (e.g., texture vs. edge) are apparent. The key feature of these figures is that the gray level intensity value is *smooth* as a function from the set of patches to $\mathbb{R}$, even when the intensity is not smooth in the original spatial coordinates.

We now describe some interesting choices for the feature maps $\mathcal{F}(I)$.

4.1.1 PATCH GRAPH

Let g_N be the set of N^2 filters $\{g_{i,j}\}_{i,j=1,\ldots,N}$, where $g_{i,j}$ is a $N \times N$ matrix with 1 in the i,j entry and 0 elsewhere. Then $\mathcal{F}_{g_N,0}$ is the set of patches of the image embedded in N^2 dimensions. The diffusion one gets from this choice of filters is the NL-means filter of Buades et al. (2005a). "NL"

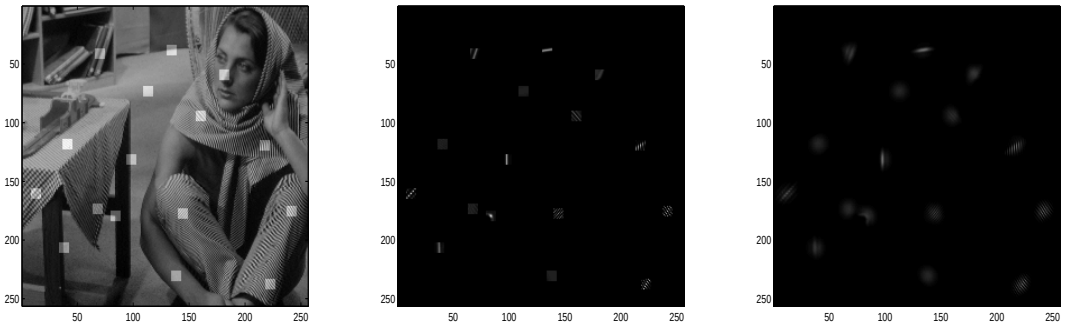


Figure 2: Left to right: image of Barbara, with several locations p_i highlighted; $K^t(p_i,\cdot)$, for $t = 1,2$.

stands for Non-Local; in the paper, they proposed setting $\alpha = 0$. In a later paper they add some locality constraints; see Buades et al. (2005b) and Mahmoudi (2005). We wish to emphasize that smoothing with the NL-means filter is not, in any sort of reasonable limit, a 2-*d* PDE; rather, it is the heat equation on the set of patches of the image!

Note the embedding into 5×5 patches is the same embedding (up to a rotation) as into 5×5 DCT coordinates, and so the weight matrices constructed from these embeddings are the same. On the other hand, if we attenuate small filter responses, the weight matrices for the two filter families will be different.

4.1.2 BOOTSTRAPPING A DENOISER; OR DENOISED IMAGES GRAPH

Different denoising methods often pick up different parts of the image structure, and create different characteristic artifacts. Suppose we have obtained denoised images $f_1,...,f_d$, from a noisy image f. To make use of the various sensitivities, and rid ourselves of the artifacts, we could embed pixels $x \in Q$ into $\mathbb{R}^{d+2}$ by $x \mapsto (\alpha x, f_1(x),...,f_d(x))$. In other words we interpret $(f_i(x))_{i=1,...,d}$ as a feature vector at x. This method is an alternative to "cycle spinning"(Coifman and Donoho, 1995), that is, simply averaging the different denoisings.

In practice, we have found that a better choice of feature vector is $f_{\sigma(1)}(x),...,f_{\sigma(d)}(x)$, where σ is a random permutation of $\{1,...,d\}$ depending on x. The idea is to mix up the artifacts from the various denoisings. Note that this would not affect standard averaging, since $\sum f_i(x) = \sum f_{\sigma(i)}$.

4.2 Image Graph Denoising

Once we have the graph W and normalized diffusion K, we use K to denoise the image. The obvious bijection from pixels to vertices in the image graph induces a correspondence between functions on pixels (such as the original image) and functions on the vertices of the graph. In particular the original image can be viewed as a function I on $G(I)$. The functions K^tI are smoothed versions of I with respect to the geometry of $G(I)$. If the graph was simply the standard grid on Q, then K would be nothing other than a discretization of the standard two-dimensional heat kernel, and K^tI would be the classical smoothing of I induced by the Euclidean two-dimensional heat kernel, associated with the classical Gaussian scale space (we refer the reader to Witkin, 1983; Koenderink,

1984; Lindeberg, 1994, and references therein). In our context K^t is associated with a scale space induced by $G(I)$, which is thus a nonlinear scale space (in the sense that it depends on the original

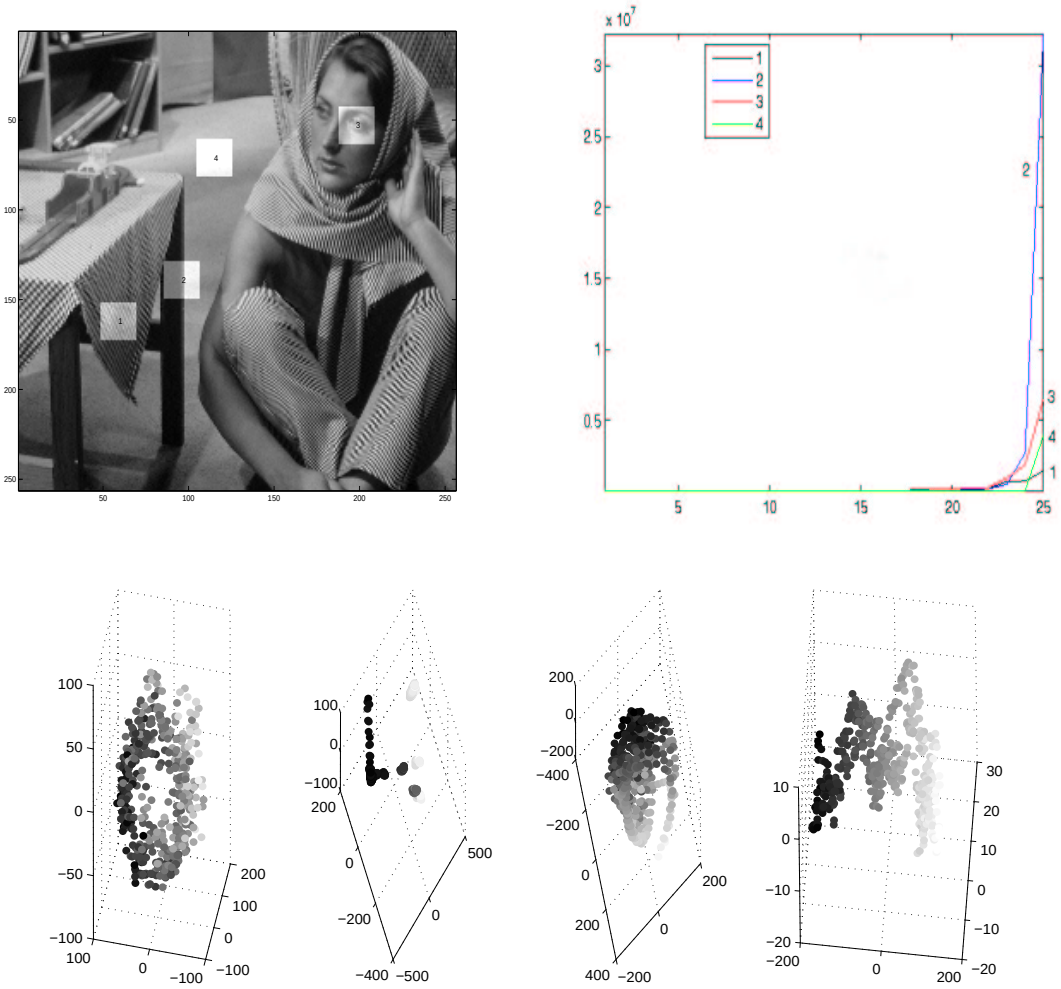


Figure 3: Top left: image of Barbara, with 4 square 10×10 pixel regions highlighted. The 5×5 patches in each region are considered as 25 dimensional vectors, and top right we plot the singular values of their covariance matrix. At the bottom, we project the 25-dimensional points in each region on their top 3 principal components, and the color is the value of the image at each point. In region 1, note how the (approximate) periodicity of the texture in region 1 is reflected in the tubular shape of the projection; in region 2, the portions of the image on different sides of the edge are disconnected in the feature space, and note the higher dimensionality, as measured by the singular values; for region 3, note the higher dimensionality (slower decay of the singular values) compared to regions 1 and 4; for region 4 note the very small dimensionality. Most importantly, note that in each region, the gray level pixel value is smooth as a function of the patches.

image I). In fact $G(I)$, as described above, is often a point cloud in high-dimensional space, where closeness in those high-dimensional space represents similarity of collections of pixels, and/or of their features, in the original two-dimensional domain of I.

We can balance smoothing by K with fidelity to the original noisy function by setting $f_{t+1} = (Kf_t + \beta f)/(1+\beta)$ where $\beta > 0$ is a parameter to be chosen, and large β corresponds to less smoothing and more fidelity to the noisy image. This is a standard technique in PDE based image processing, see Chan and Shen (2005) and references therein. If we consider iteration of K as evolving a heat equation, the fidelity term sets the noisy function as a heat source, with strength determined by β. Note that even though when we smooth in this way, the steady state is no longer the constant function, we still do not usually wish to smooth to equilibrium. We refer the reader to Figure 4 for a summary of the algorithm proposed.

$\tilde{I} \leftarrow$ `DenoiseImage`(I,t)

// Input:
// I : an image
// t : amount of denoising

// Output:
// $\tilde{I}$: a denoised version of I.

1. Construct a graph G associated with I, in any of the ways discussed in Section 4.
2. Compute the associated I-adapted diffusion operator K^I.
3. set $\tilde{I} \leftarrow (K^I)^t I$.

Figure 4: Pseudo-code for denoising an image

4.3 Examples

Figure 5 displays examples of denoising with a diffusion on an image graph. On the top left of the figure we have the noisy image f_0; the noise is $N(0,.0244)$. On the top right of Figure 5, we denoise the image using a 7×7 NL-means type patch embedding as described in Section 4.1.1. We set

$$W(k,j) = e^{-\rho_{81}^{\sim}(k,j)^2/.3}$$

where $\rho_{81}^{\sim}$ is the distance in the embedding, restricted to 81 point balls in the 2-d metric; that is we take $S(k)$ in Equation (8) to be the 81 nearest pixels to pixel k in the 2-d metric. We then normalize $K = D^{-1}W$ and denoise the image by applying K three times with a fidelity term of .07; that is, $f_{t+1} = (Kf_t + .07f_0)/(1.07)$, and the image displayed is f_3. The parameters were chosen by hand.

In the bottom row of Figure 5: on the bottom left, we sum 9 curvelet denoisings. Each curvelet denoisings is a reconstruction of the noisy image f_0 shifted either 1, 2, or 4 pixels in the vertical and/or horizontal directions, using only coefficients with magnitudes greater than 3σ. To demonstrate bootstrapping, or cycle spinning by diffusion, we embed each pixel in $\mathbb{R}^9$ using the 9 curvelet denoisings as coordinates. We set

$$W(k,j) = e^{-\rho_{81}^{\sim}(k,j)^2/.03}$$

Figure 5: 1) Lena with Gaussian noise added. 2) Denoising using a 7×7 patch graph. 3) Denoising using hard thresholding of curvelet coefficients. The image is a simple average over 9 denoisings with different grid shifts. 4) Denoising with a diffusion built from the 9 curvelet denoisings.

where $\tilde{\rho_{81}}$ is the distance in the embedding, and again we take $S(k)$ in Equation (8) to be the 81 nearest pixels to pixel k in the 2-d metric. We then normalize $K = D^{-1}W$ and denoise the image by applying K ten times with a fidelity term of .1; that is $f_{t+1} = (Kf_t + .1f_0)/(1.1)$, and f_{10} is displayed. The results are on the bottom right of Figure 5. We are able to greatly reduce the artifacts from a simple average of the curvelet denoisings.

5. Application II: Graph Transductive Learning

We apply function adapted approach to the transductive learning problem, and give experimental evidence demonstrating that using function adapted weights can improve diffusion based classifiers.

In a transductive learning problem one is given a few "labeled" examples $\tilde{X} \times \tilde{F} = \{(x_1, y_1), \ldots, (x_p, y_p)\}$ and a large number of "unlabeled" examples $X \setminus \tilde{X} = \{x_{p+1}, \ldots, x_n\}$. The goal is to estimate the conditional distributions $F(y|x)$ associated with each available example x (labeled or unlabeled).

For example $\tilde{F}$ may correspond to labels for the points $\tilde{X}$, or the result of a measurement at the points in $\tilde{X}$. The goal is to extend $\tilde{F}$ to a function F defined on the whole X, that is consistent with unseen labels/measurements at points in $X \setminus \tilde{X}$.

This framework is of interest in applications where it is easy to collect samples, that is, X is large, however it is expensive to assign a label or make a measurement at X, so only a few labels/measurements are available, namely at the points in $\tilde{X}$. The points in $X \setminus \tilde{X}$, albeit unlabeled, can be used to infer properties of the structure of the space (or underlying process/probability distribution) that is potentially useful in order to extend $\tilde{F}$ to F. Data sets with internal structures or geometry are in fact ubiquitous.

If F is smooth with respect to the data, an intrinsic analysis on the data set, such as the one possible by the use of diffusion processes and the associated Fourier and multi-scale analyses, fits very well in the transductive learning framework. In several papers a diffusion process constructed on X has been used for finding F directly (Zhou and Schlkopf, 2005; Zhu et al., 2003; Kondor and Lafferty, 2002) and indirectly, by using adapted basis functions on X constructed from the diffusion, such as the eigenfunctions of the Laplacian (Coifman and Lafon, 2006a,b; Lafon, 2004; Coifman et al., 2005a,b; Belkin and Niyogi, 2003b; Maggioni and Mhaskar, 2007), or diffusion wavelets (Coifman and Maggioni, 2006; Mahadevan and Maggioni, 2007; Maggioni and Mahadevan, 2006; Mahadevan and Maggioni, 2005; Maggioni and Mahadevan, 2005).

We will try to modify the geometry of the unlabeled data so that F is as smooth as possible with respect to the modified geometry. We will use the function adapted approach to try to learn the correct modification.

5.1 Diffusion for Classification

We consider here the case of classification, that is, F takes only a small number of values (compared to the cardinality of X), say $\{1, \ldots, k\}$. Let C_i, $i \in \{1, \ldots k\}$, be the classes, let C_i^{lab} be the labeled data points in the ith class, that is, $C_i = \{x \in \tilde{X} : \tilde{F} = i\}$, and let χ_i^{lab} be the characteristic function of those C_i, that is, $\chi_i^{lab}(x) = 1$ if $x \in C_i$, and $\chi_i^{lab}(x) = 0$ otherwise.

A simple classification algorithm can be obtained as follows (Szummer and Jaakkola, 2001):

(i) Build a geometric diffusion K on the graph defined by the data points X, as described in Section 2.2.1.

(ii) Use a power of K to smooth the functions χ_i^{lab}, exactly as in the denoising algorithm described above, obtaining functions $\overline{\chi_i^{lab}}$:

$$\overline{\chi_i^{lab}} = K^t \chi_i^{lab} .$$

The parameter t can be chosen by cross-validation.

(iii) Assign each point x to the class

$$\mathrm{argmax}_i \overline{\chi_i^{lab}}(x) .$$

This algorithm takes into account the influence of the labeled points on the unlabeled point to be classified, where the measure of influence is based on the weighted connectivity of the whole data set. If we average with a power of the kernel we have constructed, we count the number and strength of all the paths of length t to the various classes from a given data point. As a consequence, this method is more resistant to noise than, for example, a simple nearest neighbors (or also a geodesic nearest neighbors) method, as changing the location or class of a small number of data points does not change the structure of the whole network, while it can change the class label of a few nearest neighbors.

For each i, the "initial condition" for the heat flow given by χ_i^{lab} considers all the unlabeled points to be the same as labeled points not in C_i. Since we are solving many one-vs-all problems, this is reasonable; but one also may want to set the initial condition $\chi_i^{lab}(x) = 1$ for $x \in C_i^{lab}$, $\chi_i^{lab}(x) = -1$ for $x \in C_j^{lab}$, $j \neq i$, and $\chi_i^{lab}(x) = 0$ for all other x. It can be very useful to change the initial condition to a boundary condition by resetting the values of the labeled points after each application of the kernel. For large powers, this is equivalent to the harmonic classifier of Zhu et al. (2003), where the χ_i^{lab} is extended to the "harmonic" function with given boundary values on the labeled set. Just as in the image denoising examples, it is often the case that one does not want to run such a harmonic classifier to equilibrium, and we may want to find the correct number of iterations of smoothing by K and updating the boundary values by cross validation.

We can also use the eigenfunctions of K (which are also those of the Laplacian $\mathcal{L}$) to extend the classes. Belkin (2003) suggests using least squares fitting in the embedding defined by the first few eigenfunctions $\phi_1, \ldots, \phi_N$ of K. Since the values at the unlabeled points are unknown, we regress only to the labeled points; so for each χ_i^{lab}, we need to solve

$$\operatorname{argmin}_{\{a_l\}} \sum_{x \text{ labeled}} \left| \sum_{l=1}^{N} a_{il} \phi_l(x) - \chi_i^{lab}(x) \right|^2,$$

and extend the χ_i^{lab} to

$$\overline{\chi_i^{lab}} = \sum_{l=1}^{N} a_{il} \phi_i.$$

The parameter N controls the bias-variance tradeoff: smaller N implies larger bias of the model (larger smoothness)[5] and decreases the variance, while larger N has the opposite effect. Large N thus corresponds to small t in the iteration of K.

5.2 Function Adapted Diffusion for Classification

If the structure of the classes is very simple with respect to the geometry of the data set, then smoothness with respect to this geometry is precisely what is necessary to generalize from the labeled data. However, it is possible that the classes have additional structure on top of the underlying data set, which will not be preserved by smoothing geometrically. In particular at the boundaries between classes we would like to filter in such a way that the "edges" of the class function are preserved. We will modify the diffusion so it flows faster along class boundaries and slower across them, by using function-adapted kernels as in (7). Of course, we do not know the class boundaries: the func-

5. On the other hand, extending with small numbers of eigenfunctions creates "ripples"; that is, the Gibbs phenomenon. Techniques for avoiding the Gibbs phenomenon are discussed in Maggioni and Mhaskar (2007).

$F \leftarrow$ `ClassifyWithAdaptedDiffusion`$(X, \tilde{X}, \{\chi_i\}_{i=1,\ldots,N}, t_1, \beta, t_2)$

// Input:
// $X := \{x_i\}$: a data set
// $\tilde{X}$: a subset of X, representing the labeled set
// $\{\chi_i\}_{i=1,\ldots,N}$: set of characteristic functions of the classes, defined on $\tilde{X}$
// β : weight of the tuning parameter

// Output:
// C : function on X, such that $C(x)$ is the class to which $x \in X$ is estimated to belong.

1. Construct a weighted graph G associated with X, in any of the ways discussed.
2. Compute the associated diffusion operator K as in (1).
3. Compute guesses at the soft class functions $\overline{\chi_i}$ using any of the methods in Section 5.1, or any other method, and for multi-class problems, set
$$c_i(x) = \frac{\overline{\chi_i}(x)}{\sum_i |\overline{\chi_i(x)}|}.$$
4. Using the c_i as features, or $\overline{\chi_i}$ for two class problems, construct a new graph with kernel K' from the similarities as in Equation (7), with $\sigma_2 = \beta\sigma_1$.
5. Finally, find $C(x)$ using any of the methods in Section 5.1 and the kernel K'

Figure 6: Pseudo-code for learning of a function based on diffusion on graphs

tions $\{\chi_i\}$ are initially given on a (typically small) subset $\tilde{X}$ of X, and hence a similarity cannot be immediately defined in a way similar to (7).

We use a bootstrapping technique. We first use one of the algorithms above, which only uses similarities between data points ("geometry"), to generate the functions $\overline{\chi_i}$. We then use these functions to design a function-adapted kernel, by setting

$$\mathcal{F}(\{\chi_i\})(x) := (c_i(x))_{i=1,\ldots,k},$$

and then define a kernel as in (7). Here the c_i's are normalized confidence functions defined by

$$c_i(x) = \frac{\overline{\chi_i}(x)}{\sum_i |\overline{\chi_i(x)}|}.$$

In this way, if several classes claim a data point with some confidence, the diffusion will tend to average more among other points which have the same ownership situation when determining the value of a function at that data point. The normalization, besides having a clear probabilistic interpretation when the $\overline{\chi_i}$ are positive, also achieves the effect of not slowing the diffusion when there is only one possible class that a point could be in, for example, if a data point is surrounded by points of a single class, but is relatively far from all of them.

We summarize the algorithm in Figure 6. In the examples below we simply let ρ_2 be the metric of $\mathbb{R}^k$, and also let $h_2(a) = h_1(a) = e^{-a}$. The ratio β between σ_2 and σ_1, however, is important,

since it measures the trade-off between the importance given to the geometry of X and that of the set of estimates $\{(\overline{\chi_i}(x))_{i=1,\ldots,k}\}_{x\in X} \subseteq \mathbb{R}^k$.

We wish to emphasize the similarity between this technique and those described in Section 4 and especially Section 4.1.2. We allow the geometry of the data set to absorb some of the complexity of the classes, and use diffusion analysis techniques on the modified data set. The parallel with image denoising should not be unexpected: the goal of a function-adapted kernel is to strengthen averaging along level lines, and this is as desirable in image denoising as in transductive learning.

We remark that even if the c_i's are good estimates of the classes, they are not necessarily good choices for extra coordinates: for example, consider a two class problem, and a function c which has the correct sign on each class, but oscillates wildly. On the other hand, functions which are poor estimates of the classes could be excellent extra coordinates as long as they oscillate slowly parallel to the class boundaries. Our experience suggests, consistently with these considerations, that the safest choices for extra coordinates are very smooth estimates of the classes. In particular, of the three methods of class extension mentioned above, the eigenfunction method is often not a good choice for extra coordinates because of oscillation phenomena; see the examples in Section 5.4.

5.3 Relationship Between Our Methods and Previous Work

In Coifman et al. (2005a) the idea of using the estimated classes to warp the diffusion is introduced. They suggest, for each class C_n, building the modified weight matrix $W_n(i,j) = W(i,j)\overline{\chi_n^{lab}}(i)\overline{\chi_n^{lab}}(j)$, normalizing each W_n, and using the W_n to diffuse the classes. Our approach refines and generalizes theirs, by collecting all the class information into a modification of the metric used to build the kernel, rather than modifying the kernel directly. The tradeoff between geometry of the data and geometry of the (estimated/diffused) labels is made explicit and controllable.

In Zhu et al. (2003) it is proposed to adjust the graph weights to reflect prior knowledge. However, their approach is different than the one presented here. Suppose we have a two class problem. They add to each node of the graph a "dongle" node with transition probability β, which they leave as a parameter to be determined. They then run the harmonic classifier (Zhu et al., 2003) with the confidence function (ranging from 1 to -1) from a prior classifier as the boundary conditions on all the dongle nodes. Thus their method sets a tension between the *values* of the prior classifier and the harmonic classifier. Our method does not suggest values for the soft classes based on the prior classifier; rather, it uses this information to suggest modifications to the graph weights between unlabeled points.

5.4 Examples

We present experiments that demonstrate the use of function-adapted weights for transductive classification. We find that on many standard benchmark data sets, classification rate is improved using function-adapted weights instead of "geometry only" weights in diffusion based classifiers.

We use the data sets of Chapelle et al. (2006) and the first 10,000 images in the MNIST data set. At the time this article was written, the respective data sets are available at `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.html` and `http://yann.lecun.com/exdb/mnist/`, with an extensive review of the performance of existing algorithms available at `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.pdf`, and at `http://yann.lecun.com/exdb/mnist/`.

All the data sets were reduced to 50 dimensions by principal components analysis. In addition, we smooth the MNIST images by convolving 2 times with an averaging filter (a 3×3 all ones matrix). The convolutions are necessary if we want the MNIST data set to resemble a Riemannian manifold; this is because if one takes an image with sharp edges and considers a smooth family of smooth diffeomorphisms of $[0,1] \times [0,1]$, the set of images obtained under the family of diffeomorphisms is not necessarily a (differentiable) manifold (see Donoho and Grimes 2002, and also Wakin et al. 2005). However, if the image does not have edges, then the family of morphed images is a manifold.[6]

We do the following:

x1. Choose 100 points as labeled. Each of the benchmark data sets of Chapelle et al., has 12 splits into 100 labeled and 1400 unlabeled points; we use these splits. In the MNIST data set we label points 1001 through 1100 for the first split, 1101 to 1200 for the second split, etc, and used 12 splits in total. Denote the labeled points by L, let C_i the ith class, and let χ_i^{lab} be 1 on the labeled points in the ith class, -1 on the labeled points of the other classes, and 0 elsewhere.

x2. Construct a Gaussian kernel W with k nearest neighbors, $\sigma = 1$, and normalized so the jth neighbor determines unit distance in the self tuning normalization (Equation 2), where $\{k, j\}$ is one of $\{9,4\}$, $\{13,9\}$, $\{15,9\}$, or $\{21,15\}$.

x3. Classify unlabeled points x by $\sup_i \overline{\chi_i^{lab}}(x)$, where $\overline{\chi_i^{lab}}(x)$ are constructed using the harmonic classifier with the number of iterations chosen by leave-20-out cross validation from 1 to 250. More explicitly: set $g_i^0 = \chi_i^{lab}$. Set $g_i^N(x) = (Kg_i^{N-1})(x)$ if $x \notin L$, $g_i^N(x) = 1$ if $x \in C_i \bigcap L$, and $g_i^N(x) = 0$ if $x \in L \backslash C_i$, and K is W normalized to be averaging. Finally, set $\overline{\chi_i^{lab}} = g_i^N(x)$, where N is chosen by leave-10-out cross validation between 1 and 250 (C_i and L are of course reduced for the cross validation).

x4. Classify unlabeled points x by $\sup_i \overline{\chi_i^{lab}}(x)$, where the $\overline{\chi_i^{lab}}(x)$ are constructed using least squares regression in the (graph Laplacian normalized) eigenfunction embedding, with the number of eigenfunctions cross validated; that is, for each χ_i^{lab}, we solve

$$\text{argmin}_{\{a_l\}} \sum_{x \text{ labeled}} \left| \sum_{l=1}^{N} a_{il} \phi_l(x) - \chi_i(x) \right|^2,$$

and extend the χ_i^{lab} to

$$\overline{\chi_i^{lab}} = \sum_{l=1}^{N} a_{il} \phi_i.$$

The ϕ are the eigenfunctions of $\mathcal{L}$, which is W normalized as a graph Laplacian, and N is chosen by leave-10-out cross validation.

6. For the most simple example, consider a set of $n \times n$ images where each image has a single pixel set to 1, and every other pixel set to 0. As we translate the on pixel across the grid, the difference between each image and its neighbor is in a new direction in R^{n^2}, and thus there is no reasonable tangent. The same thing is true for translates of a more complicated binary image, and translates of any image with an edge. One could complain that this is an artifact of the discrete grid, but it is easy to convince yourself that the set of translates of a characteristic function in $L^2(\mathbb{R})$ does not have a tangent anywhere- the tangent direction of the curve defined by the translates of a function is exactly the derivative of the function.

x5. Classify unlabeled points x by $\sup_i \overline{\chi_i^{lab}}(x)$, where $\overline{\chi_i^{lab}}(x)$ are constructed by smoothing χ_i^{lab} with K. More explicitly: set $g_i^0 = \chi_i^{lab}$. Set $g_i^N = W g_i^{N-1}$, where K is W normalized to be averaging; and finally, let $\overline{\chi_i^{lab}} = g_i^N(x)$, where N is chosen by leave-10-out cross validation between 1 and 250 (C_i and L are of course reduced for the cross validation).

We also classify the unlabeled points using a function-adapted kernel. Using the $\overline{\chi_i^{lab}}$ from the harmonic classifier at steady state ($N = 250$), we do the following:

x6. If the problem has more than two classes, set

$$c_i(x) = \frac{g_i^{250}(x)}{\sum_i |g_i^{250}(x)|},$$

else, set $c_i(x) = g_i^{250}(x)$

x7. Using the c_i as extra coordinates, build a new weights $\tilde{W}$. The extra coordinates are normalized to have average norm equal to the average norm of the original spatial coordinates; and then multiplied by the factor β, where β is determined by cross validation from $\{1,2,4,8\}$. The modified weights are constructed using the nearest neighbors from the original weight matrix, exactly as in the image processing examples.

x8. Use the function dependent $\tilde{K}$ to estimate the classes as in (x3).

x9. Use the function dependent $\tilde{L}$ to estimate the classes as in (x4).

x10. Use the function dependent $\tilde{K}$ to estimate the classes as in (x5).

We also repeat these experiments using the smoothed classes as an initial guess, and using the eigenfunction extended classes as initial guess. The results are reported in the Figures 7, 8, and 9. Excepting the data sets g241c, gc241n, and BCI, there is an almost universal improvement in classification rate using function-adapted weights instead of "geometry only" weights over all choices of parameters and all methods of initial soft class estimation.

In addition to showing that function adapted weights often improve classification using diffusion based methods, the results we obtain are very competitive and in many cases better than all other methods listed in the extensive comparative results presented in Chapelle et al. (2006), also available at `http://www.kyb.tuebingen.mpg.de/ssl-book/benchmarks.pdf`. In Figure 10 we attempt a comparison. For every data set, we report the performance of the best classifier (with model selection, and cross-validated performance) among all the ones considered in Chapelle et al. (2006). We also report the performance of our best classifier, among the ones we considered, corresponding to different choices of the two parameters for the self-tuning nearest-neighbor graph and initial smoothing (but with other parameters cross-validated). This comparison is unfair in many respects, for us in that we give the best choice over the two graph parameters (out of the four pairs we tested) and choice of initial class estimation (three tested), and against us considering the large number of algorithms listed in Chapelle et al. (2006). Nevertheless it demonstrates that the proposed algorithms on 3 out of 7 data sets can outperform all the algorithms considered in Chapelle et al. (2006).

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.9	2.2	2.9	2.5	2.6	2.2
USPS	4.9	4.1	5.0	4.1	4.2	3.6
BCI	45.9	45.5	44.9	44.7	47.4	48.7
g241c	31.5	31.0	34.2	32.7	23.1	41.3
COIL	14.3	12.0	13.4	11.1	16.8	15.1
gc241n	25.5	24.7	27.1	25.9	13.9	35.7
text	25.5	23.7	26.3	24.0	26.4	25.4
MNIST	9.4	8.5	9.0	7.9	9.4	8.7

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.8	2.2	2.7	2.1	2.6	2.2
USPS	5.2	4.2	5.2	4.0	4.0	3.3
BCI	47.6	47.4	45.0	45.5	48.2	48.6
g241c	30.7	31.2	33.3	32.0	21.7	31.7
COIL	17.2	16.7	16.0	15.1	21.9	19.0
gc241n	23.1	21.6	25.3	22.8	11.1	24.0
text	25.2	23.0	25.5	23.3	26.9	24.0
MNIST	10.0	9.2	10.1	8.7	9.7	8.5

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.0	2.3	2.8	2.2	2.6	1.9
USPS	5.0	4.0	5.2	3.9	3.9	3.3
BCI	48.2	48.0	45.9	46.1	47.6	47.9
g241c	30.5	30.4	32.8	31.2	21.2	29.7
COIL	18.0	17.0	16.2	15.2	22.9	19.9
gc241n	24.5	21.7	26.2	23.1	11.1	17.7
text	25.1	22.4	25.7	22.3	25.6	22.9
MNIST	10.3	9.2	10.0	8.9	9.6	8.3

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.1	2.6	2.9	2.6	2.0	2.1
USPS	5.6	4.7	5.6	4.4	4.4	3.7
BCI	48.2	48.5	46.3	46.7	48.9	48.5
g241c	28.5	28.2	32.1	29.4	18.0	23.6
COIL	19.8	19.3	19.2	17.9	26.3	24.1
gc241n	21.8	20.5	24.6	21.7	9.2	14.2
text	25.1	22.3	25.6	22.7	25.4	23.2
MNIST	10.8	10.0	10.7	9.7	10.8	10.0

Figure 7: Various classification error percentages. Each pair of columns corresponds to a smoothing method; the right column in each pair uses function adapted weights, with c_i determined by the harmonic classifier. KS stands for kernel smoothing as in (x5), FAKS for function adapted kernel smoothing as in (x10), HC for harmonic classifier as in (x3), FAHC for function adapted harmonic classifier as in (x8), EF for eigenfunctions as in (x4), and FAEF for function adapted eigenfunctions as in (x9). The Gaussian kernel had k neighbors, and the jth neighbor determined unit distance in the self-tuning construction, where counterclockwise, from the top left, $\{k, j\}$ is $\{9,4\}$, $\{13,9\}$, $\{15,9\}$, and $\{21,15\}$. Notice that excepting the data sets g241c, gc241n, and BCI, there is an almost universal improvement in classification error with function-adapted weights.

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.9	2.4	2.9	2.4	2.6	2.1
USPS	4.9	4.6	5.0	4.6	4.2	3.3
BCI	45.9	47.0	44.9	45.3	47.4	47.8
g241c	31.5	29.3	34.2	29.2	23.1	33.1
COIL	14.3	13.3	13.4	12.4	16.9	16.8
gc241n	25.5	21.3	27.1	22.5	13.9	23.0
text	25.5	24.5	26.3	25.0	26.4	24.6
MNIST	9.4	7.9	9.0	7.7	9.4	7.3

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.8	2.2	2.7	2.1	2.6	2.1
USPS	5.2	4.3	5.2	4.0	4.0	3.5
BCI	47.6	48.7	45.0	46.5	48.2	49.1
g241c	30.7	27.9	33.3	27.7	21.7	28.1
COIL	17.2	17.6	16.0	15.5	22.5	20.3
gc241n	23.1	17.9	25.3	19.3	11.1	21.0
text	25.2	23.8	25.5	23.7	26.9	24.5
MNIST	10.0	8.2	10.1	8.2	9.7	7.7

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.0	2.5	2.8	2.2	2.6	1.9
USPS	5.0	4.0	5.2	3.9	3.9	3.4
BCI	48.2	48.6	45.9	46.5	47.6	48.1
g241c	30.5	26.9	32.8	27.9	21.2	27.3
COIL	18.0	17.6	16.2	15.8	22.3	21.0
gc241n	24.5	19.7	26.2	20.8	11.1	19.5
text	25.1	22.8	25.7	23.3	25.6	23.4
MNIST	10.3	8.3	10.0	7.9	9.6	7.7

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.1	2.6	2.9	2.6	2.0	2.1
USPS	5.6	4.9	5.6	4.2	4.4	4.2
BCI	48.2	49.0	46.3	47.1	48.9	49.0
g241c	28.5	26.0	32.1	26.5	18.0	22.8
COIL	19.8	19.4	19.2	18.3	26.6	23.1
gc241n	21.8	16.5	24.6	17.4	9.2	14.3
text	25.1	22.9	25.6	23.0	25.4	22.8
MNIST	10.8	9.6	10.7	9.2	10.8	8.2

Figure 8: Various classification results, c_i determined by smoothing by K. The table is otherwise organized as in Figure 7.

6. Some Comments on the Benchmarks where Our Methods Do Not Work Well

If the class structure is trivial with respect to the geometry of the data as presented, then anisotropy will be unhelpful. This is the case for two of the benchmark data sets, g241c and g241n. In g241c, which has been constructed by generating two Gaussian clouds, and labeling each point by which cloud it came from, the best possible strategy (knowing the generative model) is to assign a point

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.9	2.9	2.9	2.6	2.6	2.4
USPS	4.9	4.1	5.0	3.8	4.2	4.1
BCI	45.9	47.1	44.9	46.0	47.4	48.7
g241c	31.5	25.3	34.2	26.7	23.1	23.7
COIL	14.3	13.0	13.4	12.0	16.5	16.6
gc241n	25.5	16.7	27.1	18.2	13.9	14.1
text	25.5	25.1	26.3	25.6	26.4	25.4
MNIST	9.4	7.4	9.0	6.9	9.4	7.9

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	2.8	2.0	2.7	2.1	2.6	2.3
USPS	5.2	3.8	5.2	3.6	4.0	3.4
BCI	47.6	48.1	45.0	46.9	48.2	48.5
g241c	30.7	23.8	33.3	24.7	21.7	21.6
COIL	17.2	17.5	16.0	15.4	22.0	21.5
gc241n	23.1	13.0	25.3	14.1	11.1	11.5
text	25.2	24.8	25.5	24.9	26.9	27.3
MNIST	10.0	7.8	10.1	7.3	9.7	7.4

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.0	2.5	2.8	2.2	2.6	2.2
USPS	5.0	4.1	5.2	3.5	3.9	3.2
BCI	48.2	47.5	45.9	45.7	47.6	47.9
g241c	30.5	23.1	32.8	24.1	21.2	21.2
COIL	18.0	17.5	16.2	16.1	22.8	22.1
gc241n	24.5	13.2	26.2	13.9	11.1	11.1
text	25.1	24.3	25.7	24.3	25.6	25.9
MNIST	10.3	8.1	10.0	7.5	9.6	8.6

	KS	FAKS	HC	FAHC	EF	FAEF
digit1	3.1	2.7	2.9	2.5	2.0	2.2
USPS	5.6	4.6	5.6	4.1	4.4	3.6
BCI	48.2	49.0	46.3	47.4	48.9	49.7
g241c	28.5	19.8	32.1	21.5	18.0	18.0
COIL	19.8	19.8	19.2	18.8	26.7	25.8
gc241n	21.8	11.0	24.6	12.0	9.2	9.2
text	25.1	24.1	25.6	24.0	25.4	24.9
MNIST	10.8	8.9	10.7	7.9	10.8	9.4

Figure 9: Various classification results, c_i determined by smoothing by eigenfunctions of $\mathcal{L}$. The table is otherwise organized as in Figure 7.

	FAKS	**FAHC**	**FAEF**	**Best of other methods**
digit1	**2.0**	**2.1**	***1.9***	2.4 (Data-Dep. Reg.)
USPS	**4.0**	**3.9**	***3.3***	4.7 (LapRLS, Disc. Reg.)
BCI	45.5	45.3	47.8	**31.4** (LapRLS)
g241c	19.8	21.5	18.0	**13.5** (Cluster-Kernel)
COIL	12.0	11.1	15.1	**9.6** (Disc. Reg.)
gc241n	11.0	12.0	9.2	**5.0** (ClusterKernel)
text	***22.3***	***22.3***	**22.8**	23.6 (LapSVM)

Figure 10: Classification errors, in percent. In the rightmost column we chose, for each data set, the best performing method with model selection, among all those discussed in Chapelle et al. (2006). In each of the remaining columns we report the performance of each of the smoothing methods described above, but with the best settings of parameters for constructing the nearest neighbor graph and type of initial class guesses, among those considered in other tables (but all other smoothing parameters, including those for the initial guesses, cross validated). The aim of this rather unfair comparison is to highlight the potential of the methods on the different data sets.

to the cluster center it is nearest to. The boundary between the classes is exactly at the bottleneck between the two clusters; in other words, the geometry/metric of the data as initially presented leads to the optimal classifier, and thus modifying the geometry by the cluster guesses can only do harm. This is clearly visible if one looks at the eigenfunctions of the data set: the sign of the second eigenfunction at a given point is an excellent guess as to which cluster that point belongs to, and in fact in our experiments, often two was the optimal number of eigenfunctions. See figure 11. g241n

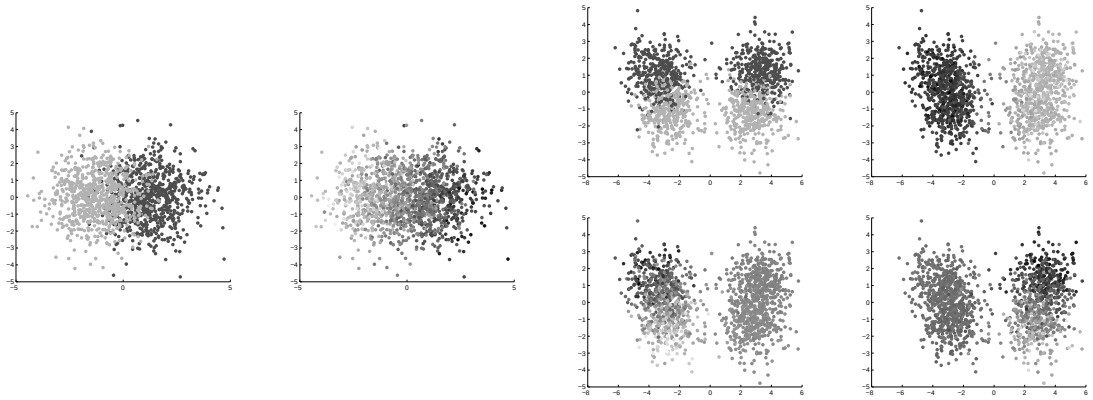


Figure 11: Panel on the left. On the left the lighter and darker points are the two classes for g241c. On the right is the second eigenfunction. Panel on the right. On the top left the lighter and darker points are the two classes for g241n. On the top right is the second eigenfunction, then on the bottom the third and fourth eigenfunctions.

is very similar; it is generated by four Gaussians. However, two pairs of centers are close together, and the pairs are relatively farther apart. The classes split across the two fine scale clusters in each coarse scale cluster as in g241c. In this data set, the ideal strategy is to decide which coarse cluster a point is in, and then the problem is exactly as above. In particular, the optimal strategy is given by the geometry of the data as presented. This is again reflected in the simplicity of the classes with respect to eigenfunctions 2, 3, and 4; see figure 11.

While in some sense these situations are very reasonable, it is our experience that in many natural problems the geometry of the data is not so simple with respect to the classes, and function-adapted kernels help build better classifiers.

Our method also was not useful for the BCI example. Here the problem was simply that the initial guess at the classes was too poor.

7. Computational Considerations

Let N be the cardinality of the data set X, which is endowed with some metric ρ. The first and most computationally intensive part of the algorithms proposed is the construction of the graph and corresponding weights. The approach we use is direct, in the sense that we explicitly store the similarity matrix W. For each point $x \in X$, we need to find the points in an ε-ball, or the k nearest neighbors of x. This problem can be solved trivially, for any metric ρ, in $O(dN^2)$ computations. It is of course highly desirable to reduce this cost, and this requires more efficient ways of computing near (or

nearest) neighbors. This problem is known to be hard even in Euclidean space $\mathbb{R}^d$, as d increases. The literature on the subject is vast, rather than a long list of papers, we point the interested reader to Datar et al. (2004) and references therein. The very short summary is that for approximate versions of the k-nearest neighbor problem, there exist algorithms which are subquadratic in N, and in fact pretty close to linear. The neighbor search is in fact the most expensive part of the algorithm: once for each point x we know its neighbors, we compute the similarities W (this is $O(k)$ for the k neighbors of each point), and create the $N \times N$ sparse matrix W (which contains kN non-zero entries). The computation of K from W is also trivial, requiring $O(N)$ with a very small constant. Apply K^t to a function f on X is very fast as well (for $t << N$, as is the case in the algorithm we propose), because of the sparsity of K, and takes $O(tkN)$ computations.

This should be compared with the $O(N^2)$ or $O(N^3)$ algorithms needed for other kernel methods, involving the computations of many eigenfunctions of the kernel, or of the Green's function $(I - K)^{-1}$.

Note that in most of the image denoising applications we have presented, because of the 2-d locality constraints we put on the neighbor searches, the number of operation is linear in the number N of pixels, with a rather small constant. In higher dimensions, for all of our examples, we use the nearest neighbor searcher provided in the TSTool package, available at `http://www.physik3.gwdg.de/tstool/`. The entire processing of an image as in the examples 256×256 takes about 7 seconds on a laptop with a 2.2Ghz dual core Intel processor (the code is not parallelized though, so it runs on one core only), and 2Gb of RAM (the memory used during processing is approximately 200Mb).

8. Future Work

We mention several directions for further study. The first one is to use a transductive learning approach to tackle image processing problems like denoising and inpainting. One has at one's disposal an endless supply of clean images to use as the "unlabeled data", and it seems that there is much to be gained by using the structure of this data.

The second one is to more closely mimic the function regularization in image processing in the context of transductive learning. In this paper, our diffusions regularize in big steps; also our method is linear (on a modified space). Even though there is no differential structure on our data sets, it seems that by using small time increments and using some sort of constrained nearest neighbor search so that we do not have to rebuild the whole graph after each matrix iteration, we can use truly nonlinear diffusions to regularize our class functions.

Another research direction is towards understanding how to construct and use efficiently basis functions which are associated to function-adapted diffusion kernels. The use of the low-frequency eigenfunctions of the operator, and the associated Fourier analysis of functions on the set has been considered in several works, as cited above, while the construction and use of multiscale basis functions, which correspond to a generalized wavelet analysis on data sets (Coifman and Maggioni, 2006; Szlam et al., 2005; Maggioni et al., 2005), has been used so far for approximation problems in machine learning (Maggioni and Mahadevan, 2006; Mahadevan and Maggioni, 2007) but has potential in many other applications. One can consider the approach that uses diffusion kernels directly, as in this paper, as a sort of "PDE approach" (even if in fact the discreteness and roughness of the sets considered usually brings us quite afar from PDEs on continua), while one can investigate "dual" approaches based on representations and bases functions.

9. Conclusions

We have introduced a general approach for associating graphs and diffusion processes to data sets and functions on such data sets. This framework is very flexible, and we have shown two particular applications, denoising of images and transductive learning, which traditionally are considered very different and have been tackled with very different techniques. We show that in fact they are very similar problems and results at least as good as the state-of-the-art can be obtained within the single framework of function-adapted diffusion kernels.

Acknowledgments

The authors would like to thank Francis Woolfe and Triet Le for helpful suggestions on how to improve the manuscript, and to James C. Bremer and Yoel Shkolnisky for developing code for some of the algorithms. MM is grateful for partial support by NSF DMS-0650413 and ONR N00014-07-1-0625 313-4224.

References

M. Belkin. *Problems of learning on manifolds*. PhD thesis, University of Chicago, 2003.

M. Belkin and P. Niyogi. Using manifold structure for partially labelled classification. *Advances in NIPS*, 15, 2003a.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14 (NIPS 2001)*, pages 585–591. MIT Press, Cambridge, 2001.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 6(15):1373–1396, June 2003b.

M. Belkin and P. Niyogi. Semi-supervised learning on Riemannian manifolds. *Machine Learning*, 56(Invited Special Issue on Clustering):209–239, 2004. TR-2001-30, Univ. Chicago, CS Dept., 2001.

Mikhail Belkin and Partha Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. In *COLT*, pages 486–500, 2005.

P. Bérard, G. Besson, and S. Gallot. Embedding Riemannian manifolds by their heat kernel. *Geom. and Fun. Anal.*, 4(4):374–398, 1994.

T. Boult, R.A. Melter, F. Skorina, and I. Stojmenovic. G-neighbors. *Proc. SPIE Conf. Vision Geom. II*, pages 96–109, 1993.

A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Model. Simul.*, 4(2):490–530 (electronic), 2005a. ISSN 1540-3459.

A. Buades, B. Coll, and J. M. Morel. Denoising image sequences does not require motion estimation. *CMLA Preprint*, (12), 2005b.

T. F. Chan and J. Shen. *Image processing and analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005. ISBN 0-89871-589-X. Variational, PDE, wavelet, and stochastic methods.

O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006. URL http://www.kyb.tuebingen.mpg.de/ssl-book.

R.T. Chin and C.L. Yeh. Quantitative evaluation of some edge-preserving noise-smoothing techniques. *Computer Vision, Graphics, and Image Processing*, 23:67–91, 1983.

F. R. K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 1997. ISBN 0-8218-0315-8.

R. R. Coifman and D. L. Donoho. Translation-invariant de-noising. Technical report, Department of Statistics, 1995. URL citeseer.ist.psu.edu/coifman95translationinvariant.html.

R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *PNAS*, 102(21):7426–7431, 2005a. doi: 10.1073/pnas.0500334102. URL http://www.pnas.org/cgi/content/abstract/102/21/7426.

R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *PNAS*, 102(21):7432–7438, 2005b. doi: 10.1073/pnas.0500334102.

R.R. Coifman and S. Lafon. Diffusion maps. *Appl. Comp. Harm. Anal.*, 21(1):5–30, 2006a.

R.R. Coifman and S. Lafon. Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions. *Appl. Comp. Harm. Anal.*, 21(1):31–52, 2006b.

R.R. Coifman and M. Maggioni. Multiscale data analysis with diffusion wavelets. *Proc. SIAM Bioinf. Workshop, Minneapolis*, April 2007. Tech. Rep. YALE/DCS/TR-1335, 2005.

R.R. Coifman and M. Maggioni. Diffusion wavelets. *Appl. Comp. Harm. Anal.*, 21(1):53–94, July 2006. (Tech. Rep. YALE/DCS/TR-1303, Yale Univ., Sep. 2004).

M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-885-7. doi: http://doi.acm.org/10.1145/997817.997857.

L.S. Davis and A. Rosenfeld. Noise cleaning by iterated local averaging. *IEEE Tran. on Systems, Man, and Cybernetics*, 8:705–710, 1978.

D. L. Donoho and C. Grimes. When does isomap recover natural parameterization of families of articulated images? Technical Report Tech. Rep. 2002-27, Department of Statistics, Stanford University, August 2002.

D. L Donoho and IM Johnstone. Ideal denoising in an orthonormal basis chosen from a library of bases. Technical report, Stanford University, 1994.

M. Elad. the origin of the bilateral filter and ways to improve it, 2002. URL citeseer.ist.psu.edu/elad02origin.html.

R.E. Graham. Snow-removal - a noise-stripping process for picture signals. *IRE Trans. on Inf. Th.*, 8:129–144, 1961.

L. Greengard and V. Rokhlin. *The rapid evaluation of potential fields in particle systems*. MIT Press, 1988.

Matthias Hein, Jean-Yves Audibert, and Ulrike von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. In *COLT*, pages 470–485, 2005.

T.S. Huang, G.J. Yang, and G.Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 27(1):13–18, 1979.

P.W. Jones, M. Maggioni, and R. Schul. Manifold parametrizations by eigenfunctions of the Laplacian and heat kernels. *Proc. Nat. Acad. Sci.*, 2007a. to appear.

P.W. Jones, M. Maggioni, and R. Schul. Universal local manifold parametrizations via heat kernels and eigenfunctions of the Laplacian. *submitted*, 2007b. http://arxiv.org/abs/0709.1975.

R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. *J. ACM*, 51(3): 497–515 (electronic), 2004. ISSN 0004-5411.

J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, Jan 1984.

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*, 2002.

S. Lafon. *Diffusion maps and geometric harmonics*. PhD thesis, Yale University, Dept of Mathematics & Applied Mathematics, 2004.

Stephane Lafon and Ann B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning and data set parameterization. *To appear in IEEE Pattern Analysis and Machine Intelligence*, to appear, 2006.

J.S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2(2):165–168, 1980.

T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 644–652, New York, NY, USA, 1998. ACM Press. ISBN 0-89791-962-9. doi: http://doi.acm.org/10.1145/276698.276880.

M. Maggioni and S. Mahadevan. Multiscale diffusion bases for policy iteration in markov decision processes. *submitted*, 2006. in preparation.

M. Maggioni and S. Mahadevan. Fast direct policy evaluation using multiscale analysis of markov diffusion processes. In *University of Massachusetts, Department of Computer Science Technical Report TR-2005-39; accepted at ICML 2006*, 2005.

M. Maggioni and H. Mhaskar. Diffusion polynomial frames on metric measure spaces. *ACHA*, 2007. in press.

M. Maggioni, J.C. Bremer Jr., R.R. Coifman, and A.D. Szlam. Biorthogonal diffusion wavelets for multiscale representations on manifolds and graphs. volume 5914, page 59141M. SPIE, 2005. URL http://link.aip.org/link/?PSI/5914/59141M/1.

S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *University of Massachusetts, Department of Computer Science Technical Report TR-2005-38; Proc. NIPS 2005*, 2005.

S. Mahadevan and M. Maggioni. Proto-value functions: A spectral framework for solving markov decision processes. *JMLR*, 8:2169–2231, 2007.

S. Mahadevan, K. Ferguson, S. Osentoski, and M. Maggioni. Simultaneous learning of representation and control in continuous domains. In *AAAI*. AAAI Press, 2006.

G. Mahmoudi, M.; Sapiro. Fast image and video denoising via nonlocal means of similar neighborhoods. *IEEE Signal Processing Letters*, 12(12):839–842, 2005.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm, 2001. URL citeseer.ist.psu.edu/ng01spectral.html.

P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):629–639, 1990.

V.C. Raykar, C. Yang, R. Duraiswami, and N. Gumerov. Fast computation of sums of gaussians in high dimensions. Technical Report CS-TR-4767, Department of Computer Science, University of Maryland, CollegePark, 2005.

L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992. ISSN 0167-2789. doi: http://dx.doi.org/10.1016/0167-2789(92)90242-F.

A. Shashua, R. Zass, and T. Hazan. Multiway clustering using supersymmetric nonnegative tensor factorization. Technical report, Hebrew University, Computer Science, Sep 2005.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Tran PAMI*, 22(8):888–905, 2000.

A. Singer. From graph to manifold Laplacian: the convergence rate. *Appl. Comp. Harm. Anal.*, 21 (1):128–134, July 2006.

R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35(2):876–879, 1964.

R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–349, 1967.

S. M. Smith and J. M. Brady. SUSAN – A new approach to low level image processing. Technical Report TR95SMS1c, Chertsey, Surrey, UK, 1995. URL citeseer.ist.psu.edu/smith95susan.html.

A. Smola and R. Kondor. Kernels and regularization on graphs, 2003. URL citeseer.ist.psu.edu/smola03kernels.html.

G. W. Soules. The rate of convergence of sinkhorn balancing. *Linear Algebra and its Applications*, 150(3):3–38, 1991.

A.D. Szlam, M. Maggioni, R.R. Coifman, and J.C. Bremer Jr. Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions. volume 5914-1, page 59141D. SPIE, 2005. URL http://link.aip.org/link/?PSI/5914/59141D/1.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001. URL citeseer.ist.psu.edu/szummer02partially.html. http://www.ai.mit.edu/people/szummer/.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proc. IEEE Inter. Conf. Comp. Vis.*, 1998.

D. Tschumperle. *PDE's Based Regularization of Multivalued Images and Applications*. PhD thesis, Universite de Nice-Sophia Antipolis, 2002.

U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. Technical Report TR-134, Max Planck Insitute for Biological Cybernetics, 2004.

M. Wakin, D. Donoho, H. Choi, and R. Baraniuk. The Multiscale Structure of Non-Differentiable Image Manifolds. In *Optics & Photonics*, San Diego, CA, July 2005.

A. P. Witkin. Scale-space filtering. In *Proc. 8th int. Joint Conf. Art. Intell.*, pages 1019–1022, 1983. Karlsruhe, Germany.

L. P. Yaroslavsky. *Digital Picture Processing*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985. ISBN 0387119345.

L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo. Weighted median filters: a tutorial. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, 43(3):155–192, 1996.

R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *International Conference on Computer Vision (ICCV)*, Oct 2005.

L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *Eighteenth Annual Conference on Neural Information Processing Systems, (NIPS)*, 2004.

H. Zha, C. Ding, M. Gu, X. He, and H.D. Simon. Spectral relaxation for *k*-means clustering. In *NIPS 2001*, pages 1057–1064. MIT Press, Cambridge, 2001.

D. Zhou and B. Schlkopf. Regularization on discrete spaces. pages 361–368, Berlin, Germany, 08 2005. Springer.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions, 2003. URL citeseer.ist.psu.edu/zhu03semisupervised.html.

Journal of Machine Learning Research 9 (2008) 1741-1755 Submitted 6/07; Revised 3/08; Published 8/08

Nearly Uniform Validation Improves Compression-Based Error Bounds

Eric Bax BAXHOME@YAHOO.COM
PO Box 60543
Pasadena, CA 91116-6543

Editor: Manfred Warmuth

Abstract

This paper develops bounds on out-of-sample error rates for support vector machines (SVMs). The bounds are based on the numbers of support vectors in the SVMs rather than on VC dimension. The bounds developed here improve on support vector counting bounds derived using Littlestone and Warmuth's compression-based bounding technique.

Keywords: compression, error bound, support vector machine, nearly uniform

1. Introduction

The error bounds developed in this paper are based on the number of support vectors in an SVM. Littlestone and Warmuth (Littlestone and Warmuth, 1986; Floyd and Warmuth, 1995) pioneered error bounds of this type. Their method derives error bounds based on how few training examples are needed to represent a classifier that is consistent with all training examples. Hence, bounds derived using their method are called compression-based bounds.

Compression-based bounds apply to SVMs because producing an SVM involves determining which training examples are "border" examples of each class and then ignoring "interior" examples. The number of border examples can be a small fraction of the number of training examples. Discarding the interior examples and training on the border examples alone produces the same SVM. So SVM training itself is a method to reconstruct the classifier based on a subset of the training data. For more details on applying compression-based bounds to SVMs, refer to Cristianini and Shawe-Taylor (2000) and von Luxburg et al. (2004). For information on applying compression-based bounds to some other classifiers, refer to Littlestone and Warmuth (1986), Floyd and Warmuth (1995), Marchand and Shawe-Taylor (2001) and Marchand and Sokolova (2005).

Compression-based bounds are effective when a small subset of the available examples can represent a classifier that is consistent with all available examples. Proofs of effectiveness for compression-based bounds use uniform validation over a set of classifiers that includes the consistent classifier. The validation is uniform in the sense that no classifier in the set may be misvalidated.

The bounds introduced in this paper apply when multiple subsets of the available examples can represent the same consistent classifier. (Support vector machines meet this condition.) Proofs of effectiveness for the new bounds use validation over a set of classifiers that includes several copies of the consistent classifier. So the validation need not be strictly uniform over the set of classifiers; the proofs can tolerate any number of misvalidated classifiers less than the number of copies of the classifier of interest and must still validate that classifier. Hence, the error bounds are said to be *nearly uniform*. Nearly uniform error bounds are introduced in Bax (1997).

This paper is organized as follows. Section 2 sets up definitions, notation, and goals. Section 3 gives an error bound for validation of a classifier. Section 4 presents a bound on the probability of several simultaneous events, which is the basis for nearly uniform error bounds. Section 5 describes nearly uniform error bounds. Section 6 applies nearly uniform error bounds to compression-based bounding. Section 7 analyzes the error bounds. Section 8 applies the error bounds. Section 9 discusses possibilities for future research.

2. Definitions, Notation, and Goals

Let $C = Z_1, \ldots, Z_m$ be a sequence of examples drawn i.i.d. from a joint input-label distribution D, with labels in $\{0,1\}$. Let $Z = (X, Y)$, where X is the input, and Y is the class label. Let g be a *classifier*, that is, a function from the input space to class labels. Define the *error* of g:

$$E_D(g) = P_D(g(X) \neq Y),$$

where the probability is over distribution D.

Let V be a sequence of examples. Define the *empirical error* of g on V:

$$E_V(g) = P_V(g(X) \neq Y),$$

where the probability is uniform over the examples in V. If a classifier has empirical error zero, then the classifier is said to be *consistent* with V.

The goal is to use the examples in C to develop a classifier g^* that is consistent with C and to produce a PAC (probably approximately correct) bound on the error. This paper focuses on producing the error bound for training methods that can develop g^* using subsets of the examples in C, called compression training algorithms. These methods include training support vector machines (SVMs) and perceptrons.

3. Validation of a Consistent Classifier

Theorem 1 *Let V be a sequence of examples drawn i.i.d. from D, and let g be a classifier developed independently of the examples in V. Then*

$$P[E_V(g) = 0 \wedge E_D(g) \geq \varepsilon] \leq (1-\varepsilon)^{|V|}.$$

Proof The LHS is

$$= P[E_V(g) = 0 | E_D(g) \geq \varepsilon] P[E_D(g) \geq \varepsilon]. \quad (1)$$

The second probability in (1) is at most one, so this is

$$\leq P[E_V(g) = 0 | E_D(g) \geq \varepsilon]. \quad (2)$$

If the error is at least ε, then the probability of correctly classifying each example in V is at most 1-ε, so (2) is

$$\leq (1-\varepsilon)^{|V|}.$$

■

The set V is called the set of *validation examples*. Theorem 1 cannot be applied directly to g^* with $V = C$ to compute an error bound, because g^* is developed using the examples in C. To validate g^*, we can use Theorem 1 indirectly, performing uniform validation over a set of classifiers that includes g^*, with validation for each classifier based on examples not used to develop the classifier. Since the set of classifiers includes g^*, uniform validation over the set implies validation of g^*.

In this paper, we use *nearly uniform* validation to validate g^*. We use a multi-set of classifiers that has several copies of g^*, and we perform validation over the classifiers, allowing fewer failed validations than the number of copies of g^*. This nearly uniform validation implies validation of g^*.

4. Probability of Several Simultaneous Events

Nearly uniform validation is based on a bound on the probability of several simultaneous events. Let $A_1, \ldots, A_n$ be subsets of a universal set U. Let $P(A_i)$ be the probability that an element drawn at random from U is a member of set A_i.

Theorem 2

$$P\left[\bigcup_{S \subseteq \{1,\ldots,n\} \wedge |S|=k} \left(\bigcap_{i \in S} A_i\right)\right] \leq \frac{1}{k}[P(A_1) + \ldots + P(A_n)],$$

that is, the probability that a random $u \in U$ is in at least k sets from $A_1, \ldots, A_n$ is at most the sum of probabilities for the sets, divided by k.

Proof The LHS of Theorem 2 is

$$P\left[I(A_1) + \ldots + I(A_n) \geq k\right], \tag{3}$$

where I is the indicator function:

$$I(A_i) = \begin{cases} 1 \text{ if } u \in A_i \\ 0 \text{ otherwise} \end{cases}.$$

By Markov's inequality, (3) is

$$\leq \frac{1}{k} E\left[I(A_1) + \ldots + I(A_n)\right].$$

By linearity of expectation, the RHS is

$$= \frac{1}{k}\left[EI(A_1) + \ldots + EI(A_n)\right],$$

which is

$$= \frac{1}{k}\left[P(A_1) + \ldots + P(A_n)\right].$$

■

Note that setting $k = 1$ gives the well-known sum bound on the probability of a union:

$$P\left[A_1 \cup \ldots \cup A_n\right] \leq P(A_1) + \ldots + P(A_n).$$

5. Nearly Uniform Validation

Consider the probability that at least k classifiers from a set of n classifiers are consistent with their validation examples and yet all have error at least ε.

Theorem 3 *Let $g_1, \ldots, g_n$ be a sequence of classifiers. Let $V_1, \ldots, V_n$ be validation sets, with each classifier g_i developed independently of validation set V_i. Let $|V| = |V_1| = \cdots = |V_n|$. Then*

$$P[\exists S \subseteq \{1,...,n\} \wedge |S| = k : \forall i \in S : (E_{V_i}(g_i) = 0 \wedge E_D(g_i) \geq \varepsilon)] \leq \frac{1}{k} n (1-\varepsilon)^{|V|},$$

where the probability is over validation sets, with the examples within each validation set drawn i.i.d. according to D, but without requiring any independence between validation sets. For instance, with a set of examples, each classifier could be the result of training on a subset of the examples, and each validation set could be the examples not used to train the corresponding classifier.

Proof We will apply Theorem 2. Define

$$\forall i \in \{1,...,n\} : A_i = \{(V_1,...,V_n) | (E_{V_i}(g_i) = 0 \wedge E_D(g_i) \geq \varepsilon)\},$$

that is, A_i is the set of validation set sequences for which g_i is consistent with V_i and yet the error of g_i is at least ε. Then the LHS of Theorem 3 is equal to the LHS of Theorem 2. So, by Theorem 2, the LHS of Theorem 3 is

$$\leq \frac{1}{k}[P(A_1) + ... + P(A_n)]. \tag{4}$$

By Theorem 1

$$\forall i \in \{1,...,n\} : P(A_i) \leq (1-\varepsilon)^{|V|}. \tag{5}$$

Substituting (5) into (4) completes the proof. ■

6. Sample Compression and Nearly Uniform Validation

This section begins with some definitions and notation. Next, Section 6.1 reviews sample compression bounds based on uniform validation. These are the compression bounds found in previous work. Then Section 6.2 develops new sample compression bounds. The new bounds are based on nearly uniform validation.

Recall that $C = Z_1, \ldots, Z_m$ is the sequence of examples available for training. For $T \subseteq \{1, \ldots, m\}$, define *g(T)* to be the classifier represented by the examples in C that are indexed by T, under some scheme for representing classifiers. (An example scheme is to train a classifier on the examples used for representation.) Define *V(T)* to be the subsequence of examples in C not indexed by T. Let

$$E_D(T) = E_D(g(T)),$$

and let

$$E_V(T) = E_{V(T)}(g(T)).$$

6.1 Review of Uniform Sample Compression Bounds

Define *compression index set H* to be a minimum-sized subset of $\{1, \ldots, m\}$ such that

$$E_V(H) = 0,$$

that is, *g(H)* is consistent with the examples in C not indexed by H. Note that any method to represent such a classifier by the examples indexed by H can be extended to represent a classifier that is consistent with all examples in C by the examples indexed by H—simply augment the classifier with the examples indexed by H, use a lookup to classify those examples correctly, and apply the original classifier to any input not in those examples. Hence, the bounds developed here also apply under the condition that H indexes a minimum-sized subset of examples in C that represent a classifier that is consistent with C.

Theorem 4 *Choose an integer $h \in \{1, \ldots, m\}$, independently of the examples in C. Identify a compression index set H. Let g*=g(H). Then*

$$P\left[E_D(g*) \geq \varepsilon \wedge |H| = h\right] \leq \left(\begin{array}{c} m \\ h \end{array} \right) (1-\varepsilon)^{m-h},$$

where the probability is over random draws of $C = Z_1, \ldots, Z_m$.

Proof Assume $|H|=h$; otherwise the probability in Theorem 4 is zero, and the proof is done. By the definition of H,

$$E_D(g*) \geq \varepsilon \Rightarrow (E_V(H) = 0 \wedge E_D(H) \geq \varepsilon).$$

So

$$P\left[E_D(g*) \geq \varepsilon\right] \leq P\left[E_V(H) = 0 \wedge E_D(H) \geq \varepsilon\right].$$

Since H depends on the examples in C, Theorem 3 does not apply directly. So use uniform validation over the set of classifiers represented by size-h subsets of C to validate $g(H)$ using Theorem 3. (This set of classifiers is chosen independently of C, and it includes $g(H)$.)

Let $g_1, \ldots, g_n$ be the classifiers represented by size-h subsets of C. Since $g(H) \in \{g_1, \ldots, g_n\}$,

$$P[E_V(H) = 0 \wedge E_D(H) \geq \varepsilon] \leq P[\exists g_i \in \{g_1, ..., g_n\} : (E_{V_i}(g_i) = 0 \wedge E_D(g_i) \geq \varepsilon)].$$

Apply Theorem 3 to the RHS. Set k=1 in Theorem 3 to bound the probability of at least one misvalidation, and note that

$$n = \left(\begin{array}{c} m \\ h \end{array} \right).$$

Then Theorem 3 implies

$$P[\exists g_i \in \{g_1, ..., g_n\} : (E_{V_i}(g_i) = 0 \wedge E_D(g_i) \geq \varepsilon)] \leq \left(\begin{array}{c} m \\ h \end{array} \right) (1-\varepsilon)^{m-h}.$$

■

In Theorem 4, we must choose h independently of C. The following theorem allows us to choose h based on C.

Theorem 5 *Let*

$$\delta(m,h,\varepsilon) = \left(\begin{array}{c} m \\ h \end{array} \right) (1-\varepsilon)^{m-h}.$$

Let $\varepsilon(m, h, \delta)$ *be the value of* ε *such that* $\delta=\delta(m, h, \varepsilon)$*:*

$$\varepsilon(m,h,\delta) = 1 - \left(\frac{\delta}{\left(\begin{array}{c} m \\ h \end{array} \right)} \right)^{\frac{1}{m-h}}.$$

Select δ*. Identify a compression index set H. Let g*=g(H). Then, with probability at least 1-*δ*,*

$$E_D(g*) \leq \varepsilon(m, |H|, \frac{\delta}{m}),$$

where the probability is over random draws of $C = Z_1, \ldots, Z_m$*.*

Proof By Theorem 4, for each $h \in \{1, \ldots, m\}$,

$$P[E_D(g*) \geq \varepsilon(m,h,\frac{\delta}{m}) \wedge h = |H|] \leq \frac{\delta}{m}.$$

Using the sum bound on the probability of a union:

$$P[\exists h \in \{1,...,m\} : E_D(g*) \geq \varepsilon(m,h,\frac{\delta}{m}) \wedge h = |H|] \leq \delta.$$

So

$$P[\forall h \in \{1,...,m\} : E_D(g*) \leq \varepsilon(m,h,\frac{\delta}{m}) \vee h \neq |H|] \geq 1-\delta.$$

■

6.2 Nearly Uniform Sample Compression Bounds for SVMs

Now consider a case where multiple subsets of the examples in C all represent the same consistent classifier. Under this condition, we can use nearly uniform validation to derive new error bounds. This section focuses on a special case of this condition, a case that applies to SVM training.

Define *retained set* $R \subseteq \{1, \ldots, m\}$ to be a minimum-sized set such that for some classifier g^*,

$$E_{V(R)}(g^*) = 0 \wedge \forall \{1,...,m\} \supseteq Q \supseteq R : g(Q) = g^*.$$

In other words, every superset of R represents the same classifier, g*, which is consistent with the examples in C not indexed by R. For example, in support vector machine training, R can be the set of support vectors in a support vector machine produced by training on all examples in C. (To ensure that the training algorithm produces the same SVM for different supersets of R, assume that the training algorithm breaks ties to determine which SVM to return in a nonrandom way that does not depend on which examples beyond R are in the training set. For example, the algorithm could

form a candidate set consisting of all SVMs with a minimum number of support vectors among those that minimize the algorithm's training objective function. Then the algorithm could return the candidate SVM with the lexicographically earliest bit-string representation.)

Theorem 6 *Choose an integer $q \in \{1, \ldots, m\}$, independently of the examples in C. Identify a retained set $R \subseteq C$ and an associated classifier g^*. Let $r=|R|$. Then*

$$P[E_D(g^*) \geq \varepsilon \wedge q \geq r] \leq \left(\begin{array}{c} m-r \\ q-r \end{array} \right)^{-1} \left(\begin{array}{c} m \\ q \end{array} \right) (1-\varepsilon)^{m-q},$$

where the probability is over random draws of $C = Z_1, \ldots, Z_m$.

Proof Assume $q = r$; otherwise the probability in Theorem 6 is zero, and the proof is done. By the definition of R,

$$E_D(g^*) \geq \varepsilon \Rightarrow \forall \{1, ..., m\} \supseteq Q \supseteq R \text{ s.t. } |Q| = q : (E_V(Q) = 0 \wedge E_D(Q) \geq \varepsilon).$$

So

$$P[E_D(g^*) \geq \varepsilon] \leq P[\forall \{1, ..., m\} \supseteq Q \supseteq R \text{ s.t. } |Q| = q : (E_V(Q) = 0 \wedge E_D(Q) \geq \varepsilon)]. \quad (6)$$

Since R depends on the examples in C, Theorem 3 does not apply directly. So use nearly uniform validation over the set of classifiers represented by size-q subsets of C to validate g^* using Theorem 3. This set of classifiers is chosen independently of C, and it includes at least k instances of g^*, where

$$k = |\{ Q | \{1, ...m\} \supseteq Q \supseteq R \wedge |Q| = q\}| = \left(\begin{array}{c} m-r \\ q-r \end{array} \right).$$

Let $g_1, \ldots, g_n$ be the classifiers represented by size-q subsets of C. Since $g_1, \ldots, g_n$ contains at least k instances of g^*, the RHS of (6) is

$$\leq P[\exists S \subseteq \{1, ..., n\} \wedge |S| = k : \forall i \in S : (E_{V_i}(g_i) = 0 \wedge E_D(g_i) \geq \varepsilon)]. \quad (7)$$

Apply Theorem 3, noting that

$$n = \left(\begin{array}{c} m \\ q \end{array} \right).$$

Then Theorem 3 implies that (7) is

$$\leq \left(\begin{array}{c} m-r \\ q-r \end{array} \right)^{-1} \left(\begin{array}{c} m \\ q \end{array} \right) (1-\varepsilon)^{m-q}.$$

■

In Theorem 6, we must choose q independently of C, and hence without reference to r. So, in Theorem 6, the value of q cannot be optimized with respect to r. Also, if $q < r$, then the theorem does not produce an error bound. The following theorem allows us to choose q based on r.

Theorem 7 *Let*

$$\delta(m,r,q,\varepsilon) = \left(\begin{array}{c} m-r \\ q-r \end{array} \right)^{-1} \left(\begin{array}{c} m \\ q \end{array} \right) (1-\varepsilon)^{m-q}.$$

Let $\varepsilon(m, r, q, \delta)$ *be the value of* ε *such that* $\delta=\delta(m, r, q, \varepsilon)$*:*

$$\varepsilon(m,r,q,\delta) = 1 - \left(\frac{\delta}{\left(\begin{array}{c} m-r \\ q-r \end{array} \right)^{-1} \left(\begin{array}{c} m \\ q \end{array} \right)} \right)^{\frac{1}{m-q}}.$$

Select δ *and a set* $W = \{q_1, \ldots, q_w\}$ *of candidates for q, independently of C. Use C to identify a retained set R and an associated classifier g*. Let* $r=|R|$*. Then, with probability at least 1-*δ*,*

$$E_D(g^*) \leq \min_{q \in W \text{ s.t. } q \geq r} \varepsilon(m,r,q,\frac{\delta}{w}),$$

where the probability is over random draws of $C = Z_1, \ldots, Z_m$*.*

Proof By Theorem 6, for each $q \in W$,

$$P[E_D(g^*) \geq \varepsilon(m,r,q,\frac{\delta}{w}) \wedge q \geq r] \leq \frac{\delta}{w}.$$

Using the sum bound on the probability of a union:

$$P[\exists q \in W : E_D(g^*) \geq \varepsilon(m,r,q,\frac{\delta}{w}) \wedge q \geq r] \leq \delta.$$

So

$$P[\forall q \in W : E_D(g^*) \leq \varepsilon(m,r,q,\frac{\delta}{w}) \vee q < r] \geq 1-\delta.$$

■

Note that setting $q = r$ and $W = \{1, \ldots, m\}$ in Theorem 7 gives the compression error bound from Theorem 5, which is the bound from the literature (Littlestone and Warmuth, 1986; Cristianini and Shawe-Taylor, 2000; Langford, 2005). In the next two sections, we examine how different choices of q and W affect the error bound.

7. Analysis

This section analyzes optimal choices of q and analyzes how strongly the error bound depends on different factors. To determine optimal choices for q, we analyze how probability of bound failure δ changes as q increases. To compare the influence of different factors, we use some approximations for the bound ε. Also, we compare choosing q to maximize the number of examples used for validation to choosing q to maximize the number of copies of g^* in the nearly uniform validation.

7.1 Optimal *q* Based on *m*, *r* and ε

In this section, we examine which values of q minimize $\delta(m,r,q,\varepsilon)$. For some background, note that increasing q increases the fraction of classifiers in the nearly uniform validation that match g^*, but it decreases the number of validation examples for each classifier. The minimum for q is r, which produces only one classifier that matches g^* and leaves m-r examples for validation. The maximum for q is m, making g^* the only classifier involved in uniform validation, but leaving no validation examples.

For fixed m, r, and ε, we want to determine values of q that minimize $\delta(m,r,q,\varepsilon)$. Let

$$p(q) = \delta(m,r,q,\varepsilon).$$

Compare values of $p(q)$ for successive values of $q \in [r,m]$, examining the ratio $p(q+1)/p(q)$. If this ratio is less than one, then increasing q improves the error bound. Writing the ratio in terms of factorials and canceling terms yields

$$p(q+1)/p(q) = (1-\frac{r}{q+1})(1-\varepsilon)^{-1}. \tag{8}$$

The RHS increases with q. So an optimal value of q is the integer that is the floor of the value that makes the RHS of (8) one. Setting the RHS equal to one and solving for q produces

$$q_{opt} = \left\lfloor \frac{r}{\varepsilon} - 1 \right\rfloor,$$

making the optimal validation set size

$$m - q_{opt} = m - \left\lfloor \frac{r}{\varepsilon} - 1 \right\rfloor.$$

For example, with SVM training, if 5% of the training examples are support vectors, and the error bound is $\varepsilon = 10\%$, then the optimal choice for q is one less than half the number of training examples.

7.2 How Error Bound ε Depends on m, r, q, and δ

To explore how the error bound $\varepsilon(m,r,q,\delta/w)$ in Theorem 7 depends on m, r, q, ,δ, and w, we will use the following pair of approximations:

$$\left(\begin{array}{c} n \\ k \end{array} \right) \approx \left(\frac{en}{k} \right)^k,$$

which follows from Stirling's approximation (Feller, 1968, p. 52), and

$$(1-a)^b \approx e^{-ab}.$$

Apply these approximations to

$$\frac{\delta}{w} = \left(\begin{array}{c} m-r \\ q-r \end{array} \right)^{-1} \left(\begin{array}{c} m \\ q \end{array} \right) (1-\varepsilon)^{m-q}, \tag{9}$$

producing

$$\frac{\delta}{w} \approx \left(\frac{e(m-r)}{q-r}\right)^{-(q-r)} \left(\frac{em}{q}\right)^{q} e^{-\varepsilon(m-q)}.$$

Solve for ε:

$$\varepsilon(m,r,q,\frac{\delta}{w}) \approx \frac{1}{m-q}\left[-(q-r)\ln\frac{e(m-r)}{q-r} + q\ln\frac{em}{q} + \ln\frac{w}{\delta}\right]. \tag{10}$$

The error bound is linear in the inverse of the number of validation examples *m* - *q*, approximately linear in *q* - *r* and in *q*, logarithmic in the number *w* of candidates for *q*, and logarithmic in the inverse of δ. (Setting *q* = *r* and *w* = *m* in (10) gives the bound from Cristianini and Shawe-Taylor 2000, p. 70.)

To compare error bounds based on uniform validation to bounds based on nearly uniform validation, compare ε(*m*,*r*,*q*,δ/*w*) with *q* = *r*, which produces a single copy of *g** in the set of classifiers being validated, to ε(*m*,*r*,*q*,δ/*w*) with *q* = (*m*+*r*)/2, which maximizes the number of copies of *g** in the set of classifiers being validated.

For *q* = *r*, use (10):

$$\varepsilon(m,r,r,\frac{\delta}{w}) \approx \frac{1}{m-r}\left[r\ln\frac{em}{r} + \ln\frac{w}{\delta}\right]. \tag{11}$$

For *q* = (*m*+*r*)/2, start from (9):

$$\frac{\delta}{w} = \begin{pmatrix} m-r \\ \frac{1}{2}(m+r)-r \end{pmatrix}^{-1} \begin{pmatrix} m \\ \frac{1}{2}(m+r) \end{pmatrix} (1-\varepsilon)^{m-(m+r)/2}.$$

Combining terms shows that this is

$$= \begin{pmatrix} m-r \\ \frac{1}{2}(m-r) \end{pmatrix}^{-1} \begin{pmatrix} m \\ \frac{1}{2}(m+r) \end{pmatrix} (1-\varepsilon)^{(m-r)/2}.$$

The first combination counts the number of copies of *g** in the set of classifiers to be validated. We chose *q* to make this the coefficient of the central (i.e., largest) term of a binomial distribution. Using the bounds for the central and near-central terms of the binomial distribution from Feller (1968, p. 180), shows this to be

$$\approx \sqrt{1-\frac{r}{m}}2^{r}e^{-(m-r)\varepsilon/2}..$$

For *r*<<*m*, the first term is close to one, so ignore it. Then

$$\frac{\delta}{w} \approx e^{r\ln 2-(m-r)\varepsilon/2}..$$

Solve for ε:

$$\varepsilon(m,r,\frac{m+r}{2},\frac{\delta}{w}) \approx \frac{2}{m-r}\left(r\ln 2 + \ln\frac{w}{\delta}\right). \tag{12}$$

Compare (11) to (12):

$$\varepsilon(m,r,r,\frac{\delta}{w}) : \varepsilon(m,r,\frac{m+r}{2},\frac{\delta}{w}) \approx \frac{1}{m-r}\left[r\ln\frac{em}{r}+\ln\frac{w}{\delta}\right] : \frac{2}{m-r}\left(r\ln 2+\ln\frac{w}{\delta}\right).$$

Terms $\ln(w/\delta)$ tend to be small compared to the rest of the sums in parentheses, so ignore them. Then divide both sides of the ratio by $r/(m\text{-}r)$ to get:

$$\approx \ln\frac{em}{r} : \ln 4,$$

which is

$$= \ln m - \ln r + 1 : \ln 4.$$

For example, if there are m = 1024 training examples and r = 64 support vectors, then the ratio is 3:1, indicating that using nearly uniform validation improves the bound by a factor of about three.

8. Tests

This section presents results of tests applying Theorem 7 to compare uniform error bounds to some nearly uniform bounds. We compare the bound methods:

1. **Uniform** – Use $q = r$ and $W = \{1, \ldots, m\}$. This is the compression-based bound from the literature.

2. **Full** – Use the optimal q in $W = \{1, \ldots, m\}$. This is the straightforward nearly uniform bound.

3. **Sample** – Use the optimal q in $W = \{m/11, 2m/11, \ldots, 10m/11\}$, that is, use 10 equally-spaced candidates for q. This limits the candidates for q, making $w = 10$ in the error bound instead of $w = m$, but optimizing over fewer choices for q.

4. **Center** – Use $q = m/2$. So $W = \{m/2\}$, and $w = 1$.

For all tests, $\delta = 0.01$, and bounds are produced by applying Theorem 7. Each table in this section shows error bounds produced by various methods for a set of problems. For each problem, the best error bound is shown in bold. In parentheses after the bounds are values of q that produced the bounds. For methods Full and Sample, q_{min} is the value of $q \in W$ that minimizes $\varepsilon(m, r, q, \delta/w)$ in Theorem 7. For the other methods, the value of q shown is the only choice.

8.1 Error Bounds for SVMs Trained on Real-World Data Sets

This subsection applies the bound methods to actual data sets for which SVMs have been developed:

1. **Netclass** – SVMs were trained to recognize which of several generative graph models best describe a graph of the neural network of c. elegans (Middendorf et al., 2004). There are m = 800 training examples and r = 51 support vectors.

2. **Genex** – SVMs were trained to classify microarray gene expression data (Brown et al., 1999). There are m = 1097 training examples and r = 216 support vectors.

				Bound Method		
Data	***m***	***r***	**Uniform (*q*)**	**Full (q_{min})**	**Sample (q_{min})**	**Center (*q*)**
Netclass	800	51	23.2% (51)	11.2% (440)	10.0% (509)	**9.8%** (400)
Genex	1097	216	46.5% (216)	25.8% (810)	**24.3%** (897)	28.1% (548)
Dig1	787	355	71.9% (355)	53.5% (648)	**52.1%** (715)	65.6% (393)

Table 1: Error Bounds for Real-World Data Sets

		Bound Method		
r	**Uniform (q)**	**Full (q_{min})**	**Sample (q_{min})**	**Center (q)**
5	25.0% (5)	19.7% (21)	17.1% (27)	**15.0%** (50)
10	35.6% (10)	27.2% (33)	24.5% (36)	**21.3%** (50)
20	50.9% (20)	39.8% (50)	36.9% (54)	**34.1%** (50)

Table 2: Error Bounds for m = 100 Examples

3. **Dig1** – An SVM was trained for digit recognition (Langford 2005). There are m = 787 training examples and r = 355 support vectors.

Method Center produces the best bound for problem Netclass, and method Sample produces the best bound for the other problems. For the first two problems (Netclass and Genex), all methods based on nearly uniform bounds produce about the same bounds, and they are about half the error bound produced by uniform validation. For Dig1, the bounds produced by methods Full and Sample are much better than those produced by uniform validation, but still not good enough to be of any use in practice.

Why are compression bounds for Dig1 so ineffective? Compression bounds are based on the idea that if a classifier is based on only a few training examples and still performs well on the rest, then that is evidence that the classifier performs well in general. For Dig1, the size of the retained set, r, is about half of the number of training examples m. The retained set is composed of training examples used in the classifier and of training examples for which the classifier errs. Consider the following scenario: each class label is equally likely, and we simply choose g^* to be the function that returns the most common label in the training set regardless of the input. Then the retained set consists of all training examples with the least common label, which is most likely a little less than half the training examples. In this case, the true error rate is 50%, and r is about half of m. Since our compression bounds are based on r and m, the bounds cannot distinguish this scenario from the case of Dig1. Hence, compression bounds rely heavily on having few retained examples relative to the number of training examples.

8.2 Error Bounds for m = 1000 Examples

This section explores error bounds produced by the different methods over a range of training set sizes m and retained set sizes r. These tests give a sense of how data set sizes and ratios of r to m affect bounds.

As in Section 8.1, the most effective bound methods in Tables 2 to 4 are Sample and Center. Comparing methods within rows shows that the nearly uniform methods produce better bounds than the uniform methods, with the nearly uniform methods producing bounds that are about half

		Bound Method		
r	**Uniform (q)**	**Full (q_{min})**	**Sample (q_{min})**	**Center (q)**
50	19.5% (50)	9.0% (480)	7.9% (636)	**7.7%** (500)
100	30.9% (100)	15.1% (620)	**13.8%** (727)	14.6% (500)
200	47.0% (200)	26.4% (742)	**24.9%** (818)	28.5% (500)

Table 3: Error Bounds for m = 1000 Examples

		Bound Method		
r	**Uniform (q)**	**Full (q_{min})**	**Sample (q_{min})**	**Center (q)**
50	11.7% (50)	4.6% (895)	4.0% (1090)	**3.9%** (1000)
100	19.2% (100)	7.7% (1209)	**7.0%** (1454)	7.3% (1000)
200	30.6% (200)	13.6% (1374)	**12.7%** (1454)	14.2% (1000)

Table 4: Error Bounds for m = 2000 Examples

the bounds for the uniform method when the ratio r:m is about 1:10. The advantage of using nearly uniform methods is more pronounced for smaller ratios of r:m.

Comparing Table 2 to Table 3 cell-by-cell shows the effect of increasing problem size by a factor of 10 while keeping ratios r:m the same. In general, the bounds improve as problem size increases, and the improvement is greater for smaller r:m ratios. The same kind of comparison is possible between Table 3 and Table 4 by comparing the first two rows of Table 3 to the last two rows of Table 4. This comparison shows the same general trends.

9. Discussion

This section outlines several possible directions for future work. One possibility is to improve the bounds by treating training examples for which g^* errs differently from training examples that comprise g^*. Right now, these examples are combined in the retained set R. Let R_E be the set of training errors for g^*, and let R^* be the set of examples used to form g^*. Suppose training on any superset of R^* yields g^*, that is, including some training errors from R_E does not disrupt training. Then R^* can be used in place of R to form a new error bound on g*. Of course, we need to use validation of non-consistent classifiers in the proposed bound, since validation sets would contain examples that cause empirical error. For example, we could use the bounds based on Binomial Tail Inversion (Langford, 2005).

The error bounds in this paper are based on uniform validation over different validation sets resulting from partitions of all available data into training and validation sets. Lack of knowledge of the joint distribution of misvalidations forces us to take the worst-case joint distributions as bases for the bounds. The worst-case bound is often applied when many validations all use the same examples; better bounds apply when the validations are all based on example sets drawn independently of each other. For each pair of partitions into training and validation sets, the validation sets have an intersection of shared examples, and the non-intersection examples are drawn independently of each other. Perhaps it is possible to use some information about the patterns of shared and independent examples among the different validation sets to constrain the joint distribution of misvalidations in a way that improves the uniform error bounds.

It would be useful to extend the results of this paper to other classifiers that have compression-based bounds, including set covering machines (SCMs) (Marchand and Shawe-Taylor 2001) and decision list machines (DLMs) (Marchand and Sokolova 2005). The challenge is to efficiently identify a retained set under the present training methods for SCMs and DLMs, that is, identify a small subset of training examples such that training on any superset that is a subset of the training examples produces the same classifier. A solution may be to modify the training algorithms in some way to make it easy to identify a small retained set after training.

An alternative approach is to empirically estimate the fraction of trainings on subsets of training data (and perhaps on strings of side information) that produce the same classifier as the classifier g^* trained on all available data. Use sampling over subsets of training data (and strings of side information) to estimate the fraction. Then form an error bound that uses the estimated fraction as the basis for nearly uniform validation. Include a term in the error bound to account for the possibility of over-estimating the fraction of trainings that produce g^*.

Finally, it should be possible to apply this empirical approach to nearly uniform validation in a transductive setting, where the inputs of examples to be classified are known. Each classifier g that agrees with g^* on all examples to be classified could be considered equivalent to g^*. This procedure is similar to empirically determining VC dimension for specific data sets, as described by Vapnik (1998).

Acknowledgments

Thanks to John Langford for extremely helpful advice, encouragement, and data. Thanks to Mario Marchand for encouragement, pointers to relevant literature, and feedback on presentation. Thanks to Manfred Warmuth and three anonymous referees for many helpful suggestions. Thanks to Lance Williams and Dan Ruderman for discussions that led to this paper and for feedback on several versions of the results. Thanks to Danny Hillis and everyone at Applied Minds for encouragement and support to pursue this research.

References

E. Bax. Similar classifiers and vc error bounds, caltechcstr:1997.cs-tr-97-14. Technical report, California Institute of Technology, 1997. Also available as `http://resolver.caltech.edu/CaltechCSTR:1997.cs-tr-97-14`.

M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, M. Ares Jr., and D. Haussler. Support vector machine classification of microarray gene expression data, ucsc-crl 99-09. Technical report, University California Santa Cruz, 1999.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

W. Feller. *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons, New York, 1968.

S. Floyd and M. Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, 21(3):1–36, 1995.

J. Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, 6:273–306, 2005.

N. Littlestone and M. Warmuth. Relating data compression and learnability, 1986. Unpublished manuscript, University of California Santa Cruz.

M. Marchand and J. Shawe-Taylor. Learning with the set covering machine. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 345–352, 2001.

M. Marchand and M. Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Research*, 6:427–451, 2005.

M. Middendorf, E. Ziv, C. Adams, J. Hom, R. Koytcheff, C. Levovitz, G. Woods, L. Chen, and C. Wiggins. Discriminative topological features reveal biological network mechanisms. *BMC Bioinformatics*, 5(181), 2004.

V. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.

U. von Luxburg, O. Bousquet, and B. Scholkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, 5:293–323, 2004.

Journal of Machine Learning Research 9 (2008) 1757-1774 Submitted 7/07; Revised 4/08; Published 8/08

Learning from Multiple Sources*

Koby Crammer CRAMMER@CIS.UPENN.EDU
Michael Kearns MKEARNS@CIS.UPENN.EDU
Jennifer Wortman WORTMANJ@SEAS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA

Editor: Peter Bartlett

Abstract

We consider the problem of learning accurate models from multiple sources of "nearby" data. Given distinct samples from multiple data sources and estimates of the dissimilarities between these sources, we provide a general theory of which samples should be used to learn models for each source. This theory is applicable in a broad decision-theoretic learning framework, and yields general results for classification and regression. A key component of our approach is the development of approximate triangle inequalities for expected loss, which may be of independent interest. We discuss the related problem of learning parameters of a distribution from multiple data sources. Finally, we illustrate our theory through a series of synthetic simulations.

Keywords: error bounds, multi-task learning

1. Introduction

We introduce and analyze a theoretical model for the problem of learning from multiple sources of "nearby" data. As a hypothetical example of where such problems might arise, consider the following scenario: For each web user in a large population, we wish to learn a classifier for what sites that user is likely to find "interesting." Assuming we have at least a small amount of labeled data for each user (as might be obtained either through direct feedback, or via indirect means such as click-throughs following a search), one approach would be to apply standard learning algorithms to each user's data in isolation. However, if there are natural and accessible measures of similarity between the interests of pairs of users (as might be obtained through their mutual labelings of common web sites), an appealing alternative is to *aggregate* the data of "nearby" users when learning a classifier for each particular user. This alternative is intuitively subject to a trade-off between the increased sample size and how different the aggregated users are.

We treat this problem in some generality and provide a bound addressing the aforementioned trade-off. In our model there are K unknown data sources, with source i generating a distinct sample S_i of n_i observations. We assume we are given only the samples S_i, and a *disparity*[1] matrix D whose entry $D(i,j)$ bounds the difference between source i and source j. Given these inputs, we wish to

*. A preliminary version of this work appeared in *Advances in Neural Information Processing Systems 19* (Crammer et al., 2007).

1. We avoid using the term distance since our results include settings in which the underlying loss measures may not be formal distances.

decide which subset of the samples S_j will result in the best model for each source i. Our framework includes settings in which the sources produce data for classification, regression, and density estimation (and more generally any additive-loss learning problem obeying certain conditions).

Our main result is a general theorem establishing a bound on the expected loss incurred by using all data sources within a given disparity of the target source. Optimization of this bound then yields a recommended subset of the data to be used in learning a model of each source. Our bound clearly expresses a trade-off between three quantities: the sample size used (which increases as we include data from more distant models), a weighted average of the disparities of the sources whose data is used, and a model complexity term. It can be applied to any learning setting in which the underlying loss function obeys an *approximate* triangle inequality, and in which the class of hypothesis models under consideration obeys uniform convergence of empirical estimates of loss to expectations. For classification problems, the standard triangle inequality holds. For regression we prove a 2-approximation to the triangle inequality. Uniform convergence bounds for the settings we consider may be obtained via standard data-independent model complexity measures such as VC dimension and pseudo-dimension, or via more recent measures such as Rademacher complexity.

Recent work by Crammer et al. (2006) examines the considerably more limited problem of learning a model when all data sources are corrupted versions of a *single, fixed* source, for instance when each data source provides noisy samples of a fixed binary function, but with varying levels of noise. In the current work, the labels on each source may be entirely unrelated to those on other source except as constrained by the bounds on disparities, requiring us to develop new techniques. Blitzer et al. (2007) study the related problem of training classifiers using multiple sources of data drawn from different *underlying* domains but labeled using identical or similar labeling functions. Wu and Dietterich (2004) study similar problems experimentally in the context of SVMs. The framework examined here can also be viewed in the context of multi-task learning, or as a type of transfer learning (Baxter, 1995; Ben-David, 2003; Maurer, 2005).

In Section 2 we introduce a decision-theoretic framework for probabilistic learning that includes classification, regression, and many other settings as special cases, and then give our multiple source generalization of this model. In Section 3 we provide our main result, which is a general bound on the expected loss incurred by using all data within a given disparity of a target source. Section 4 discusses the most simple application of this bound to binary classification using VC theory. In Section 5, we give applications of our general theory to classification and regression using Rademacher complexity, and show more generally how the theory can be applied for any Lipschitz loss function. In Section 6 we discuss how to empirically estimate the disparity matrix from data. In Section 7, we discuss the related problem of learning parameters of a distribution from multiple data sources. Finally, in Section 8, we illustrate the theory through synthetic simulations.

2. Learning Models

Before detailing our multiple-source learning model, we first introduce a standard decision-theoretic learning framework in which our goal is to find a model minimizing a generalized notion of empirical loss (Haussler, 1992). Let the *hypothesis class* $\mathcal{H}$ be a set of models (which might be classifiers, real-valued functions, densities, etc.), and let f be the *target model*, which may or may not lie in the class $\mathcal{H}$. Let z be a (generalized) data point or observation. For instance, in (noise-free) classification and regression, z will consist of a pair $\langle x,y\rangle$ where $y = f(x)$. We assume that the target model f induces some underlying distribution P_f over observations z. In the case of classification

or regression, P_f is induced by drawing the inputs x according to some underlying distribution P, and then setting $y = f(x)$ (possibly corrupted by noise).

Each setting we consider has an associated *loss function* $\mathcal{L}(h,z)$. For example, in classification we typically consider the 0/1 loss: $\mathcal{L}(h,\langle x,y\rangle) = 0$ if $h(x) = y$, and 1 otherwise. In regression we might consider the squared loss function $\mathcal{L}(h,\langle x,y\rangle) = (y-h(x))^2$. In each case, we are interested in the expected loss of a model g_2 on target g_1, $e(g_1,g_2) = \mathrm{E}_{z\sim P_{g_1}}[\mathcal{L}(g_2,z)]$. Expected loss is not necessarily symmetric.

In our multiple source model, we are presented with K distinct mutually independent samples or *sources* of data $S_1,\ldots,S_K$, and a symmetric $K\times K$ matrix D. Each source S_i contains n_i observations that are generated from a fixed and unknown model f_i, and D satisfies $\max(e(f_i,f_j),e(f_j,f_i)) \leq D(i,j)$. When D is unknown, it often can be estimated from a small amount of data; see Section 6 for more details. Our goal is to decide which sources S_j to use in order to learn the best approximation (in terms of expected loss) to each f_i.

While we are interested in accomplishing this goal for each f_i, it suffices and is convenient to examine the problem from the perspective of a fixed f_i. Thus without loss of generality let us suppose that we are given sources $S_1,\ldots,S_K$ of size $n_1,\ldots,n_K$ from models $f_1,\ldots,f_K$ such that $\varepsilon_1 \equiv D(1,1) \leq \varepsilon_2 \equiv D(1,2) \leq \cdots \leq \varepsilon_K \equiv D(1,K)$, and our goal is to learn f_1. Here we have simply taken the problem in the preceding paragraph, focused on the problem for f_1, and reordered the other models according to our estimations or their proximity to f_1. To highlight the distinguished role of the target f_1 we shall denote it f. We denote the observations in S_j by $z_1^j,\ldots,z_{n_j}^j$. In all cases we will analyze, for any $k \leq K$, the hypothesis $\hat{h}_k$ minimizing the empirical loss $\hat{e}_k(h)$ on the first k sources $S_1,\ldots,S_k$, that is

$$\hat{h}_k = \operatorname*{argmin}_{h\in\mathcal{H}} \hat{e}_k(h) = \operatorname*{argmin}_{h\in\mathcal{H}} \frac{1}{n_{1:k}} \sum_{j=1}^{k}\sum_{i=1}^{n_j} \mathcal{L}(h,z_i^j) \ ,$$

where $n_{1:k} = n_1 + \cdots + n_k$. We also denote the expected error of function h with respect to the first k sources of data as

$$e_k(h) = \mathrm{E}\left[\hat{e}_k(h)\right] = \sum_{i=1}^{k}\left(\frac{n_i}{n_{1:k}}\right) e(f_i,h).$$

3. General Theory for the Multiple Source Problem

In this section we provide the first of our main results: a general bound on the expected loss of the model minimizing the empirical loss on the nearest k sources. Optimization of this bound leads to a recommended set of sources to incorporate when learning $f = f_1$. The key ingredients needed to apply this bound are an approximate triangle inequality and a uniform convergence bound, which we define below. In the subsequent sections we demonstrate that these ingredients can indeed be provided for a variety of natural learning problems.

Definition 1 *For $\alpha \geq 1$, we say that the α-**triangle inequality** holds for a class of models $\mathcal{F}$ and expected loss function e if for all $g_1,g_2,g_3 \in \mathcal{F}$ we have*

$$e(g_1,g_2) \leq \alpha(e(g_1,g_3)+e(g_3,g_2)).$$

The parameter $\alpha \geq 1$ is a constant that depends on $\mathcal{F}$ and e.

The choice $\alpha = 1$ yields the standard triangle inequality. We note that the restriction to models in the class $\mathcal{F}$ may in some cases be quite weak—for instance, when $\mathcal{F}$ is all possible classifiers or real-valued functions with bounded range—or stronger, as in densities from the exponential family. Our results will require only that the unknown *source* models $f_1, \ldots, f_K$ lie in $\mathcal{F}$, even when our *hypothesis* models are chosen from some possibly much more restricted class $\mathcal{H} \subseteq \mathcal{F}$. For now we simply leave $\mathcal{F}$ as a parameter of the definition.

Definition 2 *A **uniform convergence bound** for a hypothesis space $\mathcal{H}$ and loss function $\mathcal{L}$ is a bound that states that for any $0 < \delta < 1$, with probability at least $1 - \delta$ for any $h \in \mathcal{H}$*

$$|\hat{e}(h) - e(h)| \leq \beta(n, \delta) \ ,$$

where $\hat{e}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h, z_i)$ for n observations $z_1, \ldots, z_n$ generated independently according to distributions $P_1, \ldots P_n$, and $e(h) = \mathrm{E}[\hat{e}(h)]$ where the expectation is taken with respect to $z_1, \ldots, z_n$. Here β is a function of the number of observations n and the confidence δ, and depends on $\mathcal{H}$ and $\mathcal{L}$.

This definition simply asserts that for every model in $\mathcal{H}$, its empirical loss on a sample of size n and the expectation of this loss will be "close" when $\beta(n, \delta)$ is small. In general the function β will incorporate standard measures of the complexity of $\mathcal{H}$, and will be a decreasing function of the sample size n, as in the classical $O(\sqrt{d/n})$ bounds of VC theory. Our bounds will be derived from the rich literature on uniform convergence. The only twist to our setting is the fact that the observations are no longer necessarily identically distributed, since they are generated from multiple sources. However, generalizing the standard uniform convergence results to this setting is mostly straightforward as we will see in the upcoming sections on applications of the bound.

We are now ready to present our general bound.

Theorem 3 *Let e be the expected loss function for loss $\mathcal{L}$, and let $\mathcal{F}$ be a class of models for which the α-triangle inequality holds with respect to e. Let $\mathcal{H} \subseteq \mathcal{F}$ be a class of hypothesis models for which there is a uniform convergence bound β for $\mathcal{L}$. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\varepsilon_i\}_{i=1}^{K}$, $\{n_i\}_{i=1}^{K}$, and $\hat{h}_k$ be defined as above. For any δ such that $0 < \delta < 1$, with probability at least $1 - \delta$, for any $k \in \{1, \ldots, K\}$*

$$e(f, \hat{h}_k) \leq \alpha^2 \min_{h \in \mathcal{H}} \{e(f, h)\} + (\alpha + \alpha^2) \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \varepsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) \ .$$

Before providing the proof, let us examine the bound of Theorem 3, which expresses a natural and intuitive trade-off. The first term in the bound is simply the *approximation error*, the residual loss that we incur by limiting our hypothesis model to fall in the restricted class $\mathcal{H}$. The second term is a weighted sum of the disparities of the $k \leq K$ models whose data is used with respect to the target model $f = f_1$. We expect this term to *increase* as we increase k to include more distant sources. The final term is determined by the uniform convergence bound. We expect this term to *decrease* with added sources due to the increased sample size. All three terms are influenced by the strength of the approximate triangle inequality that we have, as quantified by α.

The bound given in Theorem 3 can be loose, but provides an upper bound necessary for optimization and suggests a natural choice for the number of sources k^* to use to estimate the target

f:

$$k^* = \operatorname*{argmin}_k \left((\alpha+\alpha^2) \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \varepsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) \right).$$

Theorem 3 and this optimization make the implicit assumption that the best subset of sources to use will be a prefix of the sources—that is, that we should not "skip" a nearby source in favor of more distant ones. This assumption will be true for typical data-independent uniform convergence such as VC dimension bounds, and will be true on average for data-dependent bounds, where we expect uniform convergence bounds to improve with increased sample size.

We now give the proof of Theorem 3.

Proof: (Theorem 3) By Definition 1, for any $h \in \mathcal{H}$, any $k \in \{1, \ldots K\}$, and any $i \in \{1, \ldots, k\}$,

$$\left(\frac{n_i}{n_{1:k}} \right) e(f,h) \leq \left(\frac{n_i}{n_{1:k}} \right) (\alpha e(f, f_i) + \alpha e(f_i, h)) \ .$$

Summing over all $i \in \{1, \ldots, k\}$, we find

$$\begin{aligned} e(f,h) &\leq \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) (\alpha e(f, f_i) + \alpha e(f_i, h)) \\ &= \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) e(f, f_i) + \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) e(f_i, h) \leq \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \varepsilon_i + \alpha e_k(h) \ . \end{aligned}$$

In the first line above we have used the α-triangle inequality to deliberately introduce a weighted summation involving the f_i. In the second line, we have broken up the summation using the fact that $e(f, f_i) \leq \varepsilon_i$ and the definition of $e_k(h)$. Notice that the first summation is a weighted average of the expected loss of each f_i, while the second summation is the expected loss of h on the data. Using the uniform convergence bound, we may assert that with high probability $e_k(h) \leq \hat{e}_k(h) + \beta(n_{1:k}, \delta/2K)$, and with high probability

$$\hat{e}_k(\hat{h}_k) = \min_{h \in \mathcal{H}} \{\hat{e}_k(h)\} \leq \min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) e(f_i, h) + \beta(n_{1:k}, \delta/2K) \right\} \ .$$

Putting these pieces together, we find that with high probability

$$\begin{aligned} e(f, \hat{h}_k) &\leq \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \varepsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) + \alpha \min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) e(f_i, h) \right\} \\ &\leq \alpha \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \varepsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) \\ &\quad + \alpha \min_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \alpha e(f_i, f) + \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \alpha e(f, h) \right\} \\ &= (\alpha+\alpha^2) \sum_{i=1}^{k} \left(\frac{n_i}{n_{1:k}} \right) \varepsilon_i + 2\alpha\beta(n_{1:k}, \delta/2K) + \alpha^2 \min_{h \in \mathcal{H}} \{e(f,h)\} \ . \end{aligned}$$

■

4. Simple Application to Binary Classification

We demonstrate the applicability of the general theory given by Theorem 3 to several standard learning settings. As a warm-up, we begin with the most straightforward application, classification using VC bounds.

In (noise-free) binary classification, we assume that our target model is a fixed, unknown and arbitrary function f from some input set $\mathcal{X}$ to $\{0,1\}$, and that there is a fixed and unknown distribution P on the $\mathcal{X}$. Note that the distribution P over input does not depend on the target function f. The observations are of the form $z = \langle x, y\rangle$ where $y \in \{0,1\}$. The loss function $\mathcal{L}(h, \langle x, y\rangle)$ is defined as 0 if $y = h(x)$ and 1 otherwise, and the corresponding expected loss is $e(g_1, g_2) = \mathrm{E}_{\langle x,y\rangle \sim P_{g_1}}\left[\mathcal{L}(g_2, \langle x, y\rangle)\right] = \Pr_{x\sim P}[g_1(x) \neq g_2(x)]$.

For 0/1 loss it is well-known and easy to see that the (standard) 1-triangle inequality holds. Classical VC theory (Vapnik, 1998) provides us with uniform convergence as follows.

Lemma 4 *Let $\mathcal{H} : \mathcal{X} \to \{0,1\}$ be a class of functions with VC dimension d, and let $\mathcal{L}(h, \langle x, y\rangle) = |y - h(x)|$ be the 0/1-loss. The following function β is a uniform convergence bound for $\mathcal{H}$ and $\mathcal{L}$ when $n \geq d/2$:*

$$\beta(n, \delta) = \sqrt{\frac{8(d \ln(2en/d) + \ln(4/\delta))}{n}} \; .$$

The proof is analogous to the standard proof of uniform convergence using the VC Dimension (see, for example, Chapters 2–4 of Anthony and Bartlett (1999)), requiring only minor modifications to the symmetrization argument to handle the fact that the samples need not be uniformly distributed. It relies heavily on Hoeffding's inequality (Hoeffding, 1963), stated here for completeness.

Lemma 5 (Hoeffding's Inequality) *Let X be a set, $D_1, \cdots, D_m$ be probability distributions on X, and $f_1, \cdots, f_m$ be real-valued functions on X such that $f_i : X \to [a_i, b_i]$ for $i = 1, \cdots, m$. Then*

$$\Pr\left(\left|\left(\frac{1}{m}\sum_{i=1}^{m} f_i(x_i)\right) - \left(\frac{1}{m}\sum_{i=1}^{m} E_{x\sim D_i}[f_i(x)]\right)\right| \geq \varepsilon\right) \leq 2\exp\left(\frac{-2\varepsilon^2 m^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right) \; ,$$

where the probability is over the sequence of values x_i distributed according to D_i for all $i = 1, \cdots, m$.

With Lemma 4 in place, the conditions of Theorem 3 are easily satisfied, yielding the following result.

Theorem 6 *Let $\mathcal{F}$ be the set of all functions from an input set $\mathcal{X}$ into $\{0,1\}$ and let d be the VC dimension of $\mathcal{H} \subseteq \mathcal{F}$. Let e be the expected 0/1 loss. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\varepsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as above in the multi-source learning model, and assume that $n_1 \geq d/2$. For any δ such that $0 < \delta < 1$, with probability at least $1 - \delta$, for any $k \in \{1, \ldots, K\}$*

$$e(f, \hat{h}_k) \leq \min_{h\in\mathcal{H}}\{e(f,h)\} + 2\sum_{i=1}^{k}\left(\frac{n_i}{n_{1:k}}\right)\varepsilon_i + \sqrt{\frac{32\left(d\ln\left(2en_{1:k}/d\right) + \ln\left(8K/\delta\right)\right)}{n_{1:k}}} \; .$$

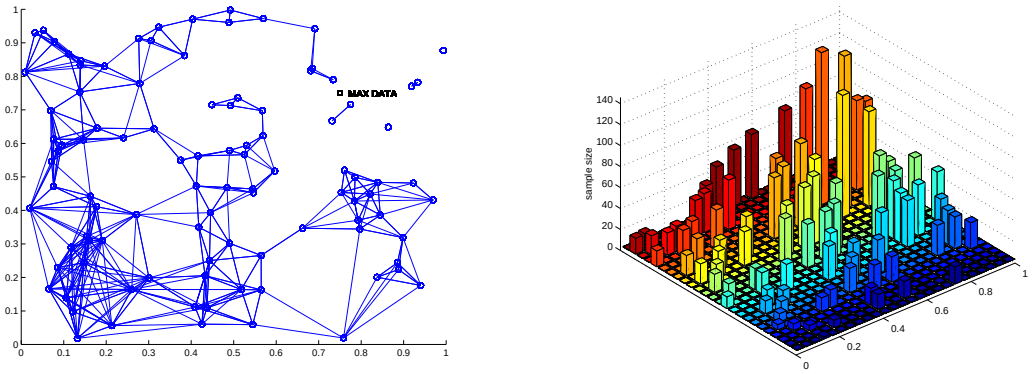


Figure 1: Visual illustration of Theorem 6.

In Figure 1 we provide a visual illustration of the behavior of Theorem 3 applied to a simple classification problem. In this problem there are $K = 100$ classifiers, each classifier f_i for $i = 1 \ldots 100$ is defined by 2 parameters represented by a point in the unit square, such that the expected disagreement rate between two such classifiers is proportional the L_1 distance between their parameters.[2] We chose the 100 parameter vectors f_i uniformly at random from the unit square (the circles in the left panel). To generate varying source sizes, we let n_i decrease with the distance of f_i from a chosen "central" point at $(0.75, 0.75)$ (marked "MAX DATA" in the left panel); the resulting source sizes for each model are shown in the bar plot in the right panel, where the origin $(0,0)$ is in the near corner, $(1,1)$ in the far corner, and the source sizes clearly peak near $(0.75, 0.75)$. For every function f_i we used Theorem 6 to find the best sources j to be used to estimate its parameters. The undirected graph on the left includes an edge between f_i and f_j if and only if the data from f_j is used to learn f_i and/or the converse.

The graph simultaneously displays the geometry implicit in Theorem 6 as well as its adaptivity to local circumstances. Near the central point, the graph is sparse and the edges quite short, corresponding to the fact that for such models we have enough direct data (represented with high bars in the right panel) that it is not advantageous to include data from distant models. Far from the central point the graph becomes dense and the edges long, as we are required to aggregate a larger neighborhood to learn the optimal model. In addition, decisions are affected locally by how many models are "nearby" a given model, when there are many close functions f_j to a given f_i there is no need to use "far" models, but when the neighborhood of a function is not populated with many examples, there is a need for data from models far-away.

5. Bounds Using Rademacher Complexity

Given the interest in tighter, potentially data-dependent convergence bounds (such as maximum margin bounds, PAC-Bayes, and others) in recent years, it is natural to ask how our multi-source theory can exploit these modern bounds. We examine one specific case here using Rademacher

2. It is easy to create simple input distributions and classifiers that generate exactly this geometry. Let the input x be a pair $x = (p,b)$ where $p \in [0,1], b \in \{0,1\}$ and let the hypothesis class consist of functions defined as pairs of thresholds $f = (t_1, t_2)$ where $f(x) = 1$ if and only if ($p > t_1$ and $b = 0$) or ($p > t_2$ and $b = 1$). The distribution of $x = (p,b)$ is a product of a uniform distribution for p and a fair coin for b.

complexity (Bartlett and Mendelson, 2002; Bartlett et al., 2002; Koltchinskii, 2001; Koltchinskii and Panchenko, 2000); analogs can be derived in a similar manner for other complexity measures. We start by deriving bounds for settings in which generic Lipschitz loss functions are used, and then discuss specific applications to classification and to regression with squared loss.

5.1 Rademacher Complexity and General Lipschitz-loss Bounds

If $\mathcal{H}$ is a class of functions mapping from a set $\mathcal{X}$ to $\mathbb{R}$, the *empirical Rademacher complexity* of $\mathcal{H}$ on a fixed set of observations $x_1,\ldots,x_n$ is defined as

$$\hat{R}_n(\mathcal{H}) = \mathrm{E}\left[\sup_{h\in\mathcal{H}}\left|\frac{2}{n}\sum_{i=1}^{n}\sigma_i h(x_i)\right|\right] ,$$

where the expectation is taken with respect to independent uniform $\{\pm 1\}$-valued random variables $\sigma_1,\ldots,\sigma_n$. The *Rademacher complexity* for n observations can then be defined as $R_n(\mathcal{H}) = \mathrm{E}\left[\hat{R}_n(\mathcal{H})\right]$ where the expectation is with respect to observations $x_1,\ldots,x_n$. In the standard setting, $x_1,\ldots,x_n$ are drawn i.i.d. from a fixed distribution. In our setting, these observations will still be independent, but not necessarily identically distributed. We will show that the standard uniform convergence results still hold for this modified definition of Rademacher complexity.

Consider any setting in which each generalized data point $z = \langle x,y\rangle$ for some $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ with $y = f(x)$. A *cost function* for the loss $\mathcal{L}$ is a function $\phi(y,a) : \mathbb{R} \to \mathbb{R}$ such that $\mathcal{L}(h,\langle x,y\rangle) = \phi(y,h(x))$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $h \in \mathcal{H}$. We will consider cost functions ϕ that are Lipschitz in the second parameter. Define $\phi'(y,a) = \phi(y,a) - \phi(y,0)$. Note that if ϕ is Lipschitz in the second parameter with constant L then ϕ' is also Lipschitz in the second parameter with the same constant L.

Lemma 8 below gives a uniform convergence bound for any loss function with a corresponding Lipschitz cost function. The proof of this lemma is in Appendix A. It is analogous to the proof of Theorem 8 in Bartlett and Mendelson (2002), which makes a similar claim in the i.i.d. setting, and uses the following lemma from Bartlett and Mendelson (2002).

Lemma 7 *If $\phi : \mathbb{R} \to \mathbb{R}$ is Lipschitz with constant L and $\phi(0) = 0$, then $R_n(\phi \circ \mathcal{H}) \leq 2LR_n(\mathcal{H})$.*

Lemma 8 *Let $\mathcal{L}$ be a loss function bounded in $[0,1]$, and $\phi : \mathbb{R} \to \mathbb{R}$ a cost function such that $\mathcal{L}(f,\langle x,y\rangle) = \phi(y,f(x))$ where ϕ is Lipschitz in the second parameter with constant L. Let $\mathcal{H} : \mathcal{X} \to \mathcal{Y}$ be a class of functions and let $\{\langle x_i,y_i\rangle\}_{i=1}^n$ be sampled independently according to some probability distributed P. For any n, for any $0 < \delta < 1$, with probability $1-\delta$ over samples of length n, every $h \in \mathcal{H}$ satisfies*

$$\beta(n,\delta) = 2LR_n(\mathcal{H}) + \sqrt{\frac{2\ln(2/\delta)}{n}} .$$

5.2 Application to Classification Using Rademacher Complexity

Theorem 9 below follows from the application of Theorem 3 using the 1-triangle inequality and an application of Lemma 8 with

$$\phi(y,a) = \begin{cases} 1 & \text{if } ya \leq 0, \\ 1-ya & \text{if } 0 < ya \leq 1, \\ 0 & \text{if } ya > 1. \end{cases}$$

Notice first that if $\mathcal{L}$ is the 0/1 loss, then for all $x \in \mathcal{X}$, $y \in \{-1,1\}$, and $h \in \mathcal{X} \to \{-1,1\}$, $\mathcal{L}(h,\langle x,y\rangle) = \phi(y,h(x))$, and furthermore that ϕ is Lipschitz with constant 1, so Lemma 8 can be applied immediately.

Theorem 9 *Let $\mathcal{F}$ be a set of functions from an input set $\mathcal{X}$ into {-1,1} and let $R_{n_{1:k}}(\mathcal{H})$ be the Rademacher complexity of $\mathcal{H} \subseteq \mathcal{F}$ on the first k sources of data. Let e be the expected 0/1 loss. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\varepsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as in the multi-source learning model. For any δ such that $0 < \delta < 1$, with probability at least $1-\delta$, for any $k \in \{1,\ldots,K\}$*

$$e(f,\hat{h}_k) \leq \min_{h\in\mathcal{H}}\{e(f,h)\} + 2\sum_{i=1}^{k}\left(\frac{n_i}{n_{1:k}}\right)\varepsilon_i + 2\sqrt{\frac{2\ln(4K/\delta)}{n_{1:k}}} + 4R_{n_{1:k}}(\mathcal{H}) \; .$$

Before moving on, let us briefly examine the behavior of this bound. Similarly to the VC-based bound given in Theorem 6, as k increases and more sources of data are combined, the second term will grow while the third will shrink. The behavior of the final term $R_{n_{1:k}}(\mathcal{H})$, however, is less predictable and may grow or shrink as more sources of data are combined.

Note that for the special case of classification with 0/1 loss, it is possible to get tighter bounds with better dependence on $R_{n_{1:k}}$ by using a more careful analysis than the one in the proof of Lemma 8. Such bounds are given in an earlier version of this paper (Crammer et al., 2007); we choose not to present these alternate bounds here to simplify presentation.

5.3 Regression

We now turn to (noise-free) regression with squared loss. Here our target model f is any function from an input class $\mathcal{X}$ into some bounded subset of $\mathbb{R}$. (Frequently we will have $\mathcal{X} \subseteq \mathbb{R}^d$, but this is not required.) Our loss function is $\mathcal{L}(h,\langle x,y\rangle) = (y-h(x))^2$, and the expected loss is thus $e(g_1,g_2) = \mathrm{E}_{\langle x,y\rangle\sim P_{g_1}}\left[\mathcal{L}(g_2,\langle x,y\rangle)\right] = \mathrm{E}_{x\sim P}\left[(g_1(x)-g_2(x))^2\right]$.

For regression it is known that the standard 1-triangle inequality does not hold. However, a 2-triangle inequality does hold and is stated in the following lemma.

Lemma 10 *Given any three functions $g_1, g_2, g_3 : \mathcal{X} \to \mathbb{R}$, a fixed and unknown distribution P on the inputs $\mathcal{X}$, and the expected loss $e(g_1,g_2) = \mathrm{E}_{x\sim P}\left[(g_1(x)-g_2(x))^2\right]$,*

$$e(g_1,g_2) \leq 2\left(e(g_1,g_3)+e(g_3,g_1)\right).$$

Proof: By Jensen's inequality and the convexity of $x \mapsto x^2$, for any g_1, g_2, and g_3,

$$\begin{aligned} e(g_1,g_2) &= \mathrm{E}_{x\sim P}\left[(g_1(x)-g_2(x))^2\right] \\ &= \mathrm{E}_{x\sim P}\left[4\left(\frac{1}{2}(g_1(x)-g_3(x)) + \frac{1}{2}(g_3(x)-g_2(x))\right)^2\right] \\ &\leq \mathrm{E}_{x\sim P}\left[2(g_1(x)-g_3(x))^2 + 2(g_3(x)-g_2(x))^2\right] = 2\left(e(g_1,g_3)+e(g_3,g_1)\right) \; . \end{aligned}$$

■

We can derive a uniform convergence bound for squared loss using Rademacher complexity as long as the region $\mathcal{Y}$ is bounded.

Lemma 11 *Let $\mathcal{H} : \mathcal{X} \to [-B,B]$ be a class of functions, and let $\mathcal{L}(h,\langle x,y\rangle) = (y-h(x))^2$ be the squared loss. The following function β is a uniform convergence bound for $\mathcal{H}$ and $\mathcal{L}$:*

$$\beta(n,\delta) = 8BR_n(\mathcal{H}) + 4B^2\sqrt{\frac{2\ln(2/\delta)}{n}}\ .$$

Proof: We cannot apply Lemma 8 directly using the squared loss function, since it may output values outside of the range $[0,1]$. Instead, we apply the Lemma 8 using the alternate loss function $\mathcal{L}'(h,\langle x,y\rangle) = \phi(y,h(x))$ where

$$\phi(y,a) = \begin{cases} \frac{1}{4B^2}(y+B)^2 & \text{if } a < -B, \\ \frac{1}{4B^2}(y-a)^2 & \text{if } -B \le a \le B, \\ \frac{1}{4B^2}(y+B)^2 & \text{if } a > B. \end{cases}$$

It is easy to see that ϕ always outputs values in the range $[0,1]$. Furthermore, for any $y \in [-B,B]$, ϕ is Lipschitz in the second parameter with parameter $1/B$. For any $[a,b] \in [-B,B]$,

$$\begin{aligned} |\phi(y,a)-\phi(y,b)| &= \frac{1}{4B^2}\left|(y-a)^2-(y-b)^2\right| = \frac{1}{4B^2}\left|a^2-b^2+2y(b-a)\right| \\ &\le \frac{1}{4B^2}\left|a^2-b^2\right| + \frac{1}{2B^2}\left|y(a-b)\right| \\ &\le \frac{1}{4B^2}\left|a+b\right|\left|a-b\right| + \frac{1}{2B^2}\left|y(a-b)\right| \le \frac{1}{B}\left|a-b\right|\ . \end{aligned}$$

Applying Lemma 8 gives a uniform convergence bound of $(2/B)R_n(\mathcal{H}) + \sqrt{2\ln(2/\delta)/n}$ for $\mathcal{L}'$. Scaling by $4B^2$ yields the bound for $\mathcal{L}$. ■

Combining this with Lemma 10 and applying Theorem 3 yields the following.

Theorem 12 *Let $\mathcal{F}$ be the set of functions from $\mathcal{X}$ into $[-B,B]$, and $\mathcal{H} \subseteq \mathcal{F}$. Let e be the expected squared loss. Let K, $f = f_1, f_2, \ldots, f_K \in \mathcal{F}$, $\{\varepsilon_i\}_{i=1}^K$, $\{n_i\}_{i=1}^K$, and $\hat{h}_k$ be defined as in the multi-source learning model. For any δ such that $0 < \delta < 1$, with probability at least $1-\delta$, for any $k \in \{1,\ldots,K\}$*

$$e(f,\hat{h}_k) \le 4\min_{h\in\mathcal{H}}\{e(f,h)\} + 6\sum_{i=1}^{k}\left(\frac{n_i}{n_{1:k}}\right)\varepsilon_i + 32BR_{n_{1:k}}(\mathcal{H}) + 16B^2\sqrt{\frac{2\ln(4K/\delta)}{n_{1:k}}}\ .$$

5.4 Remarks on the Use of Data-Dependent Complexity Measures

The following lemma, which relates the true Rademacher complexity of a function class to its empirical Rademacher complexity, follows directly from Theorem 11 of Bartlett and Mendelson (2002), the proof of which does not require samples to be identically distributed.

Lemma 13 *Let $\mathcal{H}$ be a class of functions mapping to $[-1,1]$. For any integer n, for any $0 < \delta < 1$, with probability $1-\delta$,*

$$\left|R_n(\mathcal{H}) - \hat{R}_n(\mathcal{H})\right| \le \sqrt{\frac{8\ln(2/\delta)}{n}}.$$

This lemma immediately allows us to replace $R_n(\mathcal{H})$ with that data-dependent quantity $\hat{R}_n$ in any of the bounds above for only a small penalty.

While the use of data-dependent complexity measures can be expected to yield more accurate bounds and thus better decisions about the number k^* of sources to use, it is not without its costs in comparison to the more standard data-independent approaches. In particular, in principle the optimization of a data-dependent version of the bound given in Theorem 9 to choose k^* may actually involve running the learning algorithm on all possible prefixes of the sources, since we cannot know the data-dependent complexity term for each prefix without doing so. In contrast, the data-independent bounds can be computed and optimized for k^* without examining the data at all, and the learning performed only once on the first k^* sources. This is especially useful in the case that labels are not free but must be purchased at a price.

6. Estimating the Disparity Matrix

A potential drawback of the theory presented here is the need to estimate the disparity matrix D when it is unknown. However, it is often the case that this matrix can be estimated with many fewer labeled samples than are required for learning. In this section, we discuss how D can be estimated in the classification setting.

As before, consider the scenario in which each target function is a fixed, unknown and arbitrary function from some input set $\mathcal{X}$ to $\{-1,1\}$, and assume that there is a fixed and unknown distribution P over $\mathcal{X}$. Suppose we are given m data points labeled by a pair of functions f_i and f_j, and let $\hat{e}(f_i,f_j)$ be the fraction of points on which the labels disagree. By Hoeffding's inequality, with probability $1-\delta'$,

$$|\hat{e}(f_i,f_j)-e(f_i,f_j)| \leq \sqrt{\frac{\ln(2/\delta')}{2m}}\ .$$

Thus in order to approximate $e(f_i,f_j)$ with an error no more than ε, only $\ln(2/\delta')/(2\varepsilon^2)$ commonly labeled points are needed. Applying the union bound gives us the following lemma.

Lemma 14 *Let $\mathcal{F}$ be a set of functions from $\mathcal{X}$ into $\{-1,1\}$, and suppose $f_1,\ldots,f_K \in \mathcal{F}$. Let e be the expected 0/1 loss. Suppose that for each pair $i,j \in \{1,\cdots,K\}$, there exist $m_{i,j} \geq m_0$ examples distributed according to P commonly labeled by f_i and f_j, where*

$$m_0 = \frac{2\ln(K)+\ln(2/\delta)}{2\varepsilon^2}$$

for any δ such that $0 \leq \delta \leq 1$, and let $\hat{e}(f_i,f_j)$ be the fraction of commonly labeled examples on which f_i and f_j disagree. Then with probability $1-\delta$, for all $i,j \in \{1,\cdots,K\}$, $|\hat{e}(f_i,f_j)-e(f_i,f_j)| \leq \varepsilon$.

Using the lemma we set the upper bound on the mutual error $e(f_i,f_j)$ between the pair of function f_i and f_j to be $D_{i,j} = \hat{e}(f_i,f_j)+\varepsilon$. With probability at least $1-\delta$ these bound holds simultaneously for all i,j.

Note that in general, $\log(K)$ will be significantly smaller than the dimension d of $\mathcal{H}$. Thus many fewer labeled examples are required to estimate the disparity matrix than to actually learn the best function in the class.

The assumption that there exist commonly labeled points for each pair of functions is natural in many settings. Consider, for example, the problem of predicting whether or not users will enjoy

certain movies using ratings from other users. It is often the case that pairs of users will have seen many of the same movies. These commonly rated movies can be used to determine how similar each pair of users are, while ratings of additional movies can be reserved to learn the prediction functions.

7. Estimating the Parameters of a Distribution

We now proceed with the study of the related problem of estimating the unknown parameters of a distribution from multiple sources of data. As in the previous sections, we provide a bound on the diversity of an estimator based on the first k sources from the target. Up until this point, we have measured the diversity between two functions by using the expected value of a loss function. The loss is a function of two *specific* observations. Thus, although two functions may not agree on many points, the diversity between them could be zero (if the measure of their disagreement points is zero). In this section we use a more direct way to measure the diversity between two functions by computing the distance between the parameters used to specify these distributions.

Before stating the problem formally we provide with some illustrative examples for intuition.

Example 1 *We wish to estimate the bias θ of a coin given K sources of training observations $N_1,...,N_K$. Each source N_k contains n_k outcomes of flips of a coin with bias θ_k. The only information we are given is that $\theta_k \in [\theta-\varepsilon_k, \theta+\varepsilon_k]$.*

In the next example we consider the simple generalization to the multinomial distribution, which involves more than a single parameter.

Example 2 *We wish to estimate the probability $\Theta^{(p)}$ of a die to fall on its pth side (out of D possible outcomes) given K sources of training observations $N_1,...,N_K$. Each source N_k contains n_k outcomes using a die with parameters $\Theta_k^{(p)}$. The only information provided is a bound on the ℓ_∞ distance between the parameter sets, $\max_p |\Theta_k^{(p)} - \Theta^{(p)}| = \|\Theta_k - \Theta\|_\infty \leq \varepsilon_k$.*

Formally, let $\Pr[X|\Theta]$ be a parametric family of distributions such that $X \in \mathbb{R}^d$ and $\Theta \in \mathbb{R}^D$. We assume that there exists a vector function Ψ such that

$$\mathrm{E}\left[\Psi^{(p)}(X)\right] = \Theta^{(p)} \qquad \text{for } p = 1,\ldots,D\,.$$

This assumption is met, for example, by any member of the exponential family. In the two examples we have discussed, the function Ψ is simply an identity or indicator. This function is useful because it allows us to estimate the parameters of the distribution from data. Let $X_1,\cdots,X_n$ be a sequence of n i.i.d. samples from such a distribution, where the function Ψ is known. Then the estimator obtained by the method of moments is given by the empirical mean

$$\hat{\Theta} = \frac{1}{n}\sum_{i=1}^{n}\Psi(X_i)\,.$$

In our setting, we wish to estimate the parameters Θ of a parametric distribution $\Pr[X|\Theta]$ given K sources of training observations $N_1,...,N_K$. Each source N_k contains n_k outcomes from a distribution with parameters Θ_k, that is, $\Pr[X|\Theta_k]$. The only information we are given is a bound on the ℓ_∞ distance between the parameter sets, $\|\Theta - \Theta_k\|_\infty \leq \varepsilon_k$.

We first bound the deviation of this estimation from the true parameters using Hoeffding's inequality. Fix the value of the index $p = 1,\ldots,D$. We assume that there exist A and $B > 0$ such that,

$$\Psi^{(p)}(X_i) \in [A, A+B] \quad \text{for } i = 1,\ldots,n\ .$$

Then,

$$\Pr\left[\left|\mathrm{E}\left[\hat{\Theta}^{(p)}\right] - \hat{\Theta}^{(p)}\right| \geq \varepsilon\right] \leq 2\exp\left(-2\frac{n\varepsilon^2}{B^2}\right)\ .$$

Setting the right hand-side of the inequality equal to δ and solving for ε, we get

$$\Pr\left[\left|\mathrm{E}\left[\hat{\Theta}^{(p)}\right] - \hat{\Theta}^{(p)}\right| \geq \sqrt{\frac{B^2\ln(\frac{2}{\delta})}{2n}}\right] \leq \delta\ .$$

We can use the union bound to bound on this difference for all D parameters at once and get

$$\Pr\left[\exists p\ :\ \left|\mathrm{E}\left[\hat{\Theta}^{(p)}\right] - \hat{\Theta}^{(p)}\right| \geq \sqrt{\frac{B^2\ln(\frac{2D}{\delta})}{2n}}\right] \leq \sum_{p=1}^{D}\frac{\delta}{D} = \delta\ .$$

This proves the following lemma.

Lemma 15 *Let $X_1,\ldots,X_n$ be a sequence of i.i.d. random variables. Let $\hat{\Theta} = \frac{1}{n}\sum_{i=1}^{n}\Psi(X_i)$ and $\Theta = \mathrm{E}\left[\hat{\Theta}\right]$, where both $\hat{\Theta}$ and Θ are D-dimensional vectors. Assume that $\Psi^{(p)}(X_i) \in [A, A+B]$ for $i = 1,\ldots,n$, $p = 1,\ldots,D$, for some A and $B > 0$. Then, for any $\delta \in (0,1)$ the following bound holds.*

$$\Pr\left[\|\Theta - \hat{\Theta}\|_\infty \geq \sqrt{\frac{B^2\ln(\frac{2D}{\delta})}{2n}}\right] \leq \delta\ .$$

We now turn our attention to the problem of choosing the best sources. We define the estimator using the first k sources to be,

$$\hat{\Theta}_k = \frac{1}{n_{1:k}}\sum_{i=1}^{k}\sum_{X\in N_i}\Psi(X)\ ,$$

where as before $n_{1:k} = \sum_{i=1}^{k} n_i$. We denote the expectation of this estimate by

$$\bar{\Theta}_k = \mathrm{E}\left[\hat{\Theta}_k\right] = \frac{1}{n_{1:k}}\sum_{i=1}^{k} n_i\Theta_i\ .$$

We now bound the deviation of the estimate $\hat{\Theta}_k$ from the true set of parameters Θ using the expectation $\bar{\Theta}_k$,

$$\begin{aligned}
\|\Theta - \hat{\Theta}_k\|_\infty &= \|\Theta - \bar{\Theta}_k + \bar{\Theta}_k - \hat{\Theta}_k\|_\infty \\
&\leq \|\Theta - \bar{\Theta}_k\|_\infty + \|\bar{\Theta}_k - \hat{\Theta}_k\|_\infty \\
&\leq \sum_{i=1}^{k}\frac{n_i\|\Theta - \Theta_i\|_\infty}{n_{1:k}} + \|\bar{\Theta}_k - \hat{\Theta}_k\|_\infty \\
&\leq \sum_{i=1}^{k}\frac{n_i}{n_{1:k}}\varepsilon_k + \|\bar{\Theta}_k - \hat{\Theta}_k\|_\infty\ .
\end{aligned}$$

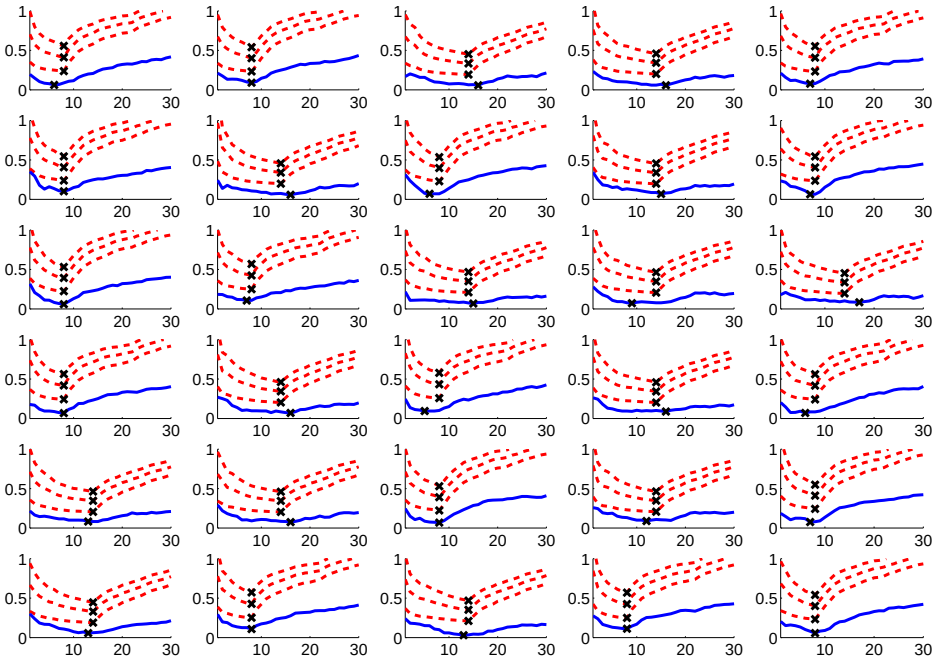


Figure 2: Simulation of the multiple source error bounds.

Let $B = \max_{k=1\ldots K} \sup_X \|\Psi(X)\|_\infty$. We can then use Lemma 15 to bound the second term above, yielding the following theorem.

Theorem 16 *Let $\hat{\Theta}_k$ be the estimate of Θ obtained by using only the data from the first k sources, where both $\hat{\Theta}$ and Θ are D-dimensional vectors. Assume that $-B \leq \Psi^{(p)}(X_i) \leq B$. Then for any $\delta > 0$, with probability $\geq 1-\delta$ we have*

$$\|\Theta - \hat{\Theta}_k\|_\infty \leq \sum_{i=1}^{k} \frac{n_i}{n_{1:k}} \varepsilon_i + \sqrt{\frac{4B^2 \ln(\frac{2DK}{\delta})}{2n_{1:k}}}$$

simultaneously for all $k = 1, \ldots, K$.

As we did with Theorem 3, we can convert Theorem 16 into an algorithm for selecting data sources. Given the K sources of data we simply compute the bounds provided by these theorems for each prefix of the sources of length k and select the subset of sources that yields the smallest bound. A bound for the special case of Example 1 was developed and presented in previous work (Crammer et al., 2006). That bound has the same form as the bound given here in Theorem 16 but with better constants.

8. Synthetic Simulations

In this section, we illustrate the bounds of our main theorem through a simple synthetic simulation. Our hypothesis class $\mathcal{H}$ consists of all linear separators through the origin in 15 dimensions. The

goal is to learn thirty classifiers from this class using only limited amounts of data. These data points are drawn uniformly at random from inside the 15-dimensional unit sphere. In this restricted setting, it is easy to calculate the disparity between two functions. Representing each function f by a unit weight vector w such that $f(x) = \text{sign}(w \cdot x)$, the distance between functions w and w' is simply θ/π where $\theta = \arccos(w \cdot w')$ is the angle between w and w'.

In each simulation we ran, the linear classifiers were generated as follows. First, three base classifiers were generated by choosing weight vectors uniformly at random from the surface of the 15-dimensional sphere. Each of the thirty classifiers was then generated by randomly choosing one of the base classifiers, perturbing each coordinate of its weight vector, and renormalizing the perturbed weights.

The number of training samples available for each function was generated from a Poisson distribution with a mean of 8. Each data instance was then sampled from inside the 15-dimensional unit sphere via rejection sampling and labeled by the corresponding classifier, and 500 test samples for each function were generated in the same manner.

To predict the optimal set of training data sources to use for each model, we calculated an approximation of the multiple-source VC bound for classification. It is well known that the constants in the VC-based uniform convergence bounds are not tight. Thus for the purpose of illustrating how these bounds might be used in practice, we have chosen to show approximations of our bounds with a variety of constants. In particular, we have chosen to approximate the bound with

$$2\sum_{i=1}^{k}\left(\frac{n_k}{n_{1:K}}\right)\varepsilon_k + C\sqrt{\frac{(d\ln(2en_{1:K}/d)+\ln(8K/\delta))}{n_{1:K}}}$$

with $\delta = 0.001$ for different values of C. These approximations yield curves that are closer in shape and magnitude to the actual error than a curve generated using the precise, overly conservative constants of Theorem 6.

The set of plots shown in Figure 2 illustrates the results of a single multiple source simulation. (Results from repeated versions of this experiment and experiments with different source sizes were similar.) Each individual plot represents a particular target function. On the x axis is the number of data sources used in training. On the y axis is error. The solid blue curves show test error of a model trained using logistic regression. Dashed red curves show our multiple source error bound with C set to $1/4$ in the lowest curve, $1/2$ in the middle curve, and $1/\sqrt{2}$ in the highest curve. The $\times$ on each curve marks the minimum value.

These plots clearly show the trade-off that exists. When too few sources are used, there is not enough data available to learn a 15-dimensional function. When too many sources are used, the labels on the training data often will not correspond to the labels that would have been assigned by the target function. The optimal amount of data lies somewhere in between.

Although the VC bounds remain loose even after constants have been dropped, the bounds tend to maintain the appropriate shape and thus predict the optimal set of sources quite well. In general, when C is set to small values, the predicted error values for small amounts of data (low k) tend to be quite accurate, while predicted values for larger amounts of data overestimate the true error. As C is set to larger values, the predictions become much larger in magnitude than the true error curves, but the shape of the prediction curves become more similar to the true error. In both cases, although the bounds are loose, they can still prove useful in determining the optimal set of sources to consider.

Acknowledgments

We thank the anonymous reviewers for many valuable suggestions, especially on the simplified presentation of the application to regression.

Appendix A. Proof of Lemma 8

The proof relies on McDiarmid's inequality (McDiarmid, 1989), which is stated here for completeness.

Lemma 17 (McDiarmid's inequality) *Let $x_1,\ldots,x_n$ be independent random variables taking on values in a set A and assume that $f : A^n \to \mathbb{R}$ satisfies*

$$\sup_{x_1,\ldots,x_n,x_i' \in A} |f(x_1,\ldots,x_n) - f(x_1,\ldots,x_{i-1},x_{i'},x_{i+1},\ldots,x_n)| \leq c_i$$

for every $1 \leq i \leq n$. Then for every $t > 0$,

$$\Pr[f(x_1,\ldots,x_n) - \mathrm{E}[f(x_1,\ldots,x_n)] \geq t] \leq \exp^{-2t^2/\Sigma_{i=1}^n c_i^2} .$$

Here we show one direction of the bound, namely that with probability $1-\delta/2$, for all $h \in \mathcal{H}$,

$$e(h) \leq \hat{e}(h) + 2LR_n(\mathcal{H}) + \sqrt{\frac{2\ln(2/\delta)}{n}} .$$

The proof of the other direction is nearly identical. For $i \in \{1,\ldots,n\}$, let $\langle x_i, y_i\rangle$ be the ith training instance, distributed according to P_i, and let $\langle x_i', y_i'\rangle$ be independent random variables drawn according to P_i. Note that for all $h \in \mathcal{H}$,

$$\begin{aligned}
e(h) &= e(h) + \hat{e}(h) - \hat{e}(h) \leq \hat{e}(h) + \sup_{h' \in \mathcal{H}} \left(e(h') - \hat{e}(h')\right) \\
&= \hat{e}(h) + \sup_{h' \in \mathcal{H}} \left(\mathrm{E}_{\{\langle x_i', y_i'\rangle\}_{i=1}^n} \left[\frac{1}{n}\sum_{i=1}^n \phi(y_i', h'(x_i')) \right] - \frac{1}{n}\sum_{i=1}^n \phi(y_i, h'(x_i)) \right) \\
&= \hat{e}(h) + \sup_{h' \in \mathcal{H}} \left(\mathrm{E}_{\{\langle x_i', y_i'\rangle\}_{i=1}^n} \left[\frac{1}{n}\sum_{i=1}^n \phi'(y_i', h'(x_i')) + \phi(y_i', 0) \right] \right. \\
&\qquad \left. - \frac{1}{n}\sum_{i=1}^n \phi'(y_i, h'(x_i)) + \phi(y_i, 0) \right) .
\end{aligned}$$

When only one instance $\langle x_i, y_i\rangle$ changes, the sup term can change by at most $2/n$. Thus we can apply McDiarmid's inequality to see that with probability at least $1-\delta/2$,

$$e(h) \leq \hat{e}(h) + \mathrm{E}\left[\sup_{h' \in \mathcal{H}} \left(\mathrm{E}\left[\frac{1}{n}\sum_{i=1}^n \phi'(y_i', h'(x_i')) \right] - \frac{1}{n}\sum_{i=1}^n \phi'(y_i, h'(x_i)) \right) \right] + \sqrt{\frac{2\ln(2/\delta)}{n}} ,$$

where the outer expectation is with respect to set of training instances $\{\langle x_i, y_i\rangle\}_{i=1}^n$ and the inner expectation is with respect to the set of random variables $\{\langle x_i', y_i'\rangle\}_{i=1}^n$. Now it suffices to show that

this middle term is bounded by $2LR_n(\mathcal{H})$. Using the fact that the supremum of an expectation is less than or equal to the expectation of a supremum, we find that

$$\begin{aligned}
&\mathrm{E}_{\{\langle x_i,y_i\rangle\}_{i=1}^n}\left[\sup_{h'\in\mathcal{H}}\left(\mathrm{E}_{\{\langle x_i',y_i'\rangle\}_{i=1}^n}\left[\frac{1}{n}\sum_{i=1}^n \phi'(y_i',h'(x_i'))\right]-\frac{1}{n}\sum_{i=1}^n \phi'(y_i,h'(x_i))\right)\right]\\
&\quad\leq\quad \mathrm{E}_{\{\langle x_i,y_i\rangle\}_{i=1}^n,\{\langle x_i',y_i'\rangle\}_{i=1}^n}\left[\sup_{h'\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^n\left(\phi'(y_i',h'(x_i'))-\phi'(y_i,h'(x_i))\right)\right]\\
&\quad=\quad \mathrm{E}_{\{\langle x_i,y_i\rangle\}_{i=1}^n,\{\langle x_i',y_i'\rangle\}_{i=1}^n,\{\sigma_i\}_{i=1}^n}\left[\sup_{h'\in\mathcal{H}}\frac{1}{n}\sum_{i=1}^n\sigma_i\left(\phi'(y_i',h'(x_i'))-\phi'(y_i,h'(x_i))\right)\right]\\
&\quad\leq\quad \mathrm{E}_{\{\langle x_i,y_i\rangle\}_{i=1}^n,\{\sigma_i\}_{i=1}^n}\left[\sup_{h'\in\mathcal{H}}\frac{2}{n}\sum_{i=1}^n\sigma_i\phi'(y_i,h'(x_i))\right]=R_n(\phi'\circ\mathcal{H})\ .
\end{aligned}$$

Lemma 7 implies that $R_n(\phi'\circ\mathcal{H})\leq 2LR_n(\mathcal{H})$ since ϕ is Lipschitz with parameter L. The result follows.

References

M. Anthony and P. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

P. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

P. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Machine Learning*, 48:85–113, 2002.

J. Baxter. Learning internal representations. In *Proceedings of the Eighth Annual Conference on Computational Learning Theory*, 1995.

S. Ben-David. Exploiting task relatedness for multiple task learning. In *Proceedings of the Sixteenth Annual Conference on Computational Learning Theory*, 2003.

J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 20*, 2007.

K. Crammer, M. Kearns, and J. Wortman. Learning from data of variable quality. In *Advances in Neural Information Processing Systems 18*, 2006.

K. Crammer, M. Kearns, and J. Wortman. Learning from multiple sources. In *Advances in Neural Information Processing Systems 19*, 2007.

D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.

V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. *High Dimensional Probability*, II:443–459, 2000.

A. Maurer. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6: 967–994, 2005.

C. McDiarmid. On the method of bounded differences. *Surveys in Combinatorics*, pages 148–188, 1989.

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

P. Wu and T. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

Journal of Machine Learning Research 9 (2008) 1775-1822 Submitted 9/07; Published 8/08

Exponentiated Gradient Algorithms for Conditional Random Fields and Max-Margin Markov Networks

Michael Collins* MCOLLINS@CSAIL.MIT.EDU
Amir Globerson* GAMIR@CSAIL.MIT.EDU
Terry Koo* MAESTRO@CSAIL.MIT.EDU
Xavier Carreras CARRERAS@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Peter L. Bartlett BARTLETT@CS.BERKELEY.EDU
University of California, Berkeley
Division of Computer Science and Department of Statistics
Berkeley, CA 94720, USA

Editor: John Lafferty

Abstract

Log-linear and maximum-margin models are two commonly-used methods in supervised machine learning, and are frequently used in structured prediction problems. Efficient learning of parameters in these models is therefore an important problem, and becomes a key factor when learning from very large data sets. This paper describes exponentiated gradient (EG) algorithms for training such models, where EG updates are applied to the convex dual of either the log-linear or max-margin objective function; the dual in both the log-linear and max-margin cases corresponds to minimizing a convex function with simplex constraints. We study both batch and online variants of the algorithm, and provide rates of convergence for both cases. In the max-margin case, $O(\frac{1}{\varepsilon})$ EG updates are required to reach a given accuracy ε in the dual; in contrast, for log-linear models only $O(\log(\frac{1}{\varepsilon}))$ updates are required. For both the max-margin and log-linear cases, our bounds suggest that the online EG algorithm requires a factor of n less computation to reach a desired accuracy than the batch EG algorithm, where n is the number of training examples. Our experiments confirm that the online algorithms are much faster than the batch algorithms in practice. We describe how the EG updates factor in a convenient way for structured prediction problems, allowing the algorithms to be efficiently applied to problems such as sequence learning or natural language parsing. We perform extensive evaluation of the algorithms, comparing them to L-BFGS and stochastic gradient descent for log-linear models, and to SVM-Struct for max-margin models. The algorithms are applied to a multi-class problem as well as to a more complex large-scale parsing task. In all these settings, the EG algorithms presented here outperform the other methods.

Keywords: exponentiated gradient, log-linear models, maximum-margin models, structured prediction, conditional random fields

∗. These authors contributed equally.

1. Introduction

Structured prediction problems involve learning to map inputs x to labels y, where the labels have rich internal structure, and where the set of possible labels for a given input is typically exponential in size. Examples of structured prediction problems include sequence labeling and natural language parsing. Several models that implement learning in this scenario have been proposed over the last few years, including log-linear models such as conditional random fields (CRFs, Lafferty et al., 2001), and maximum-margin models such as maximum-margin Markov networks (Taskar et al., 2004a).

For both log-linear and max-margin models, learning is framed as minimization of a regularized loss function which is convex. In spite of the convexity of the objective function, finding the optimal parameters for these models can be computationally intensive, especially for very large data sets. This problem is exacerbated in structured prediction problems, where the large size of the set of possible labels adds an additional layer of complexity. The development of efficient optimization algorithms for learning in structured prediction problems is therefore an important problem.

In this paper we describe learning algorithms that exploit the structure of the dual optimization problems for log-linear and max-margin models. For both log-linear and max-margin models the dual problem corresponds to the minimization of a convex function Q subject to simplex constraints (Jaakkola and Haussler, 1999; Lebanon and Lafferty, 2002; Taskar et al., 2004a). More specifically, the goal is to find

$$\operatorname*{argmin}_{\forall i,\, \mathbf{u}_i \in \Delta} Q(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n) \;, \tag{1}$$

where n is the number of training examples, each $\mathbf{u}_i$ is a vector of dual variables for the i'th training example, and $Q(\mathbf{u})$ is a convex function.[1] The size of each vector $\mathbf{u}_i$ is $|\mathcal{Y}|$, where $\mathcal{Y}$ is the set of possible labels for any training example. Furthermore, $\mathbf{u}_i$ is constrained to belong to the simplex of distributions over $\mathcal{Y}$, defined as:

$$\Delta = \left\{ \mathbf{p} \in \mathbb{R}^{|\mathcal{Y}|} : p_y \geq 0 \;\;,\;\; \sum_{y \in \mathcal{Y}} p_y = 1 \right\} \;. \tag{2}$$

Thus each $\mathbf{u}_i$ is constrained to form a distribution over the set of possible labels. The max-margin and log-linear problems differ only in their definition of Q.

The algorithms in this paper make use of exponentiated gradient (EG) updates (Kivinen and Warmuth, 1997) in solving the problem in Eq. 1, in particular for the cases of log-linear or max-margin models. We focus on two classes of algorithms, which we call *batch* and *online*. In the batch case, the entire set of $\mathbf{u}_i$ variables is updated simultaneously at each iteration of the algorithm; in the online case, a single $\mathbf{u}_i$ variable is updated at each step. The "online" case essentially corresponds to coordinate-descent on the dual function Q, and is similar to the SMO algorithm (Platt, 1998) for training SVMs. The online algorithm has the advantage of updating the parameters after every sample point, rather than after making a full pass over the training examples; intuitively, this should lead to considerably faster rates of convergence when compared to the batch algorithm, and indeed our experimental and theoretical results support this intuition. A different class of online algorithms consists of stochastic gradient descent (SGD) and its variants (e.g., see LeCun et al., 1998; Vishwanathan et al., 2006). In contrast to SGD, however, the EG algorithm is guaranteed to

1. In what follows we use $\mathbf{u}$ to denote the variables $\mathbf{u}_1, \ldots, \mathbf{u}_n$.

improve the dual objective at each step, and this objective may be calculated after each example without performing a pass over the entire data set. This is particularly convenient when making a choice of learning rate in the updates.

We describe theoretical results concerning the convergence of the EG algorithms, as well as experiments. Our key results are as follows:

- For the max-margin case, we show that $O(\frac{1}{\varepsilon})$ time is required for both the online and batch algorithms to converge to within ε of the optimal value of $Q(\mathbf{u})$. This is qualitatively similar to recent results in the literature for max-margin approaches (e.g., see Shalev-Shwartz et al., 2007). For log-linear models, we show convergence rates of $O(\log(\frac{1}{\varepsilon}))$, a significant improvement over the max-margin case.

- For both the max-margin and log-linear cases, our bounds suggest that the online algorithm requires a factor of n less computation to reach a desired accuracy, where n is the number of training examples. Our experiments confirm that the online algorithms are much faster than the batch algorithms in practice.

- We describe how the EG algorithms can be efficiently applied to an important class of structured prediction problems where the set of labels $\mathcal{Y}$ is exponential in size. In this case the number of dual variables is also exponential in size, making algorithms which deal directly with the $\mathbf{u}_i$ variables intractable. Following Bartlett et al. (2005), we focus on a formulation where each label y is represented as a set of "parts", for example corresponding to labeled cliques in a max-margin network, or context-free rules in a parse tree. Under an assumption that part-based marginals can be calculated efficiently—for example using junction tree algorithms for CRFs, or the inside-outside algorithm for context-free parsing—the EG algorithms can be implemented efficiently for both max-margin and log-linear models.

- In our experiments we compare the online EG algorithm to various state-of-the-art algorithms. For log-linear models, we compare to the L-BFGS algorithm (Byrd et al., 1995) and to stochastic gradient descent. For max-margin models we compare to the SVM-Struct algorithm of Tsochantaridis et al. (2004). The methods are applied to a standard multi-class learning problem, as well as to a more complex natural language parsing problem. In both settings we show that the EG algorithm converges to the optimum much faster than the other algorithms.

- In addition to proving convergence results for the definition of $Q(\mathbf{u})$ used in max-margin and log-linear models, we give theorems which may be useful when optimizing other definitions of $Q(\mathbf{u})$ using EG updates. In particular, we give conditions for convergence which depend on bounds relating the Bregman divergence derived from $Q(\mathbf{u})$ to the Kullback-Leibler divergence. Depending on the form of these bounds for a particular $Q(\mathbf{u})$, either $O(\frac{1}{\varepsilon})$ or $O(\log(\frac{1}{\varepsilon}))$ rates of convergence can be derived.

The rest of this paper is organized as follows. In Section 2, we introduce the log-linear and max-margin learning problems, and describe their dual optimization problems. Section 3 describes the batch and online EG algorithms; in Section 4, we describe how the algorithms can be efficiently applied to structured prediction problems. Section 5 then gives convergence proofs for the batch and

online cases. Section 6 discusses related work. Sections 7 and 8 give experiments, and Section 9 discusses our results.

This work builds on previous work described by Bartlett et al. (2005) and Globerson et al. (2007). Bartlett et al. (2005) described the application of the EG algorithm to max-margin parameter estimation, and showed how the method can be applied efficiently to part-based formulations. Globerson et al. (2007) extended the approach to log-linear parameter estimation, and gave new convergence proofs for both max-margin and log-linear estimation. The work in the current paper gives several new results. We prove rates of convergence for a randomized version of the EG online algorithm; previous work on EG algorithms had not given convergence rates for the online case. We also report new experiments, including experiments with the randomized strategy. Finally, the $O(\log(\frac{1}{\varepsilon}))$ convergence rates for the log-linear case are new. The results in Globerson et al. (2007) gave $O(\frac{1}{\varepsilon})$ rates for the batch algorithm for log-linear models, and did not give any theoretical rates of convergence for the online case.

2. Primal and Dual Problems for Regularized Loss Minimization

In this section we present the log-linear and max-margin optimization problems for supervised learning. For each problem, we describe the equivalent dual optimization problem, which will form the core of our optimization approach.

2.1 The Primal Problems

Consider a supervised learning setting with objects $x \in \mathcal{X}$ and labels $y \in \mathcal{Y}$.[2] In the structured learning setting, the labels may be sequences, trees, or other high-dimensional data with internal structure. Assume we are given a function $\mathbf{f}(x,y) : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ that maps (x,y) pairs to feature vectors. Our goal is to construct a linear prediction rule

$$h(x,\mathbf{w}) = \operatorname*{argmax}_{y \in \mathcal{Y}} \mathbf{w} \cdot \mathbf{f}(x,y) \ ,$$

with parameters $\mathbf{w} \in \mathbb{R}^d$, such that $h(x,\mathbf{w})$ is a good approximation of the *true* label of x. The parameters $\mathbf{w}$ are learned by minimizing a regularized loss

$$\mathcal{L}(\mathbf{w}; \{(x_i,y_i)\}_{i=1}^n, C) = \sum_{i=1}^n \ell(\mathbf{w},x_i,y_i) + \frac{C}{2}\|\mathbf{w}\|^2 \ ,$$

defined over a labeled training set $\{(x_i,y_i)\}_{i=1}^n$. Here $C > 0$ is a constant determining the amount of regularization. The function ℓ measures the loss incurred in using $\mathbf{w}$ to predict the label of x_i, given that the true label is y_i.

In this paper we will consider two definitions for $\ell(\mathbf{w},x_i,y_i)$. The first definition, originally introduced by Taskar et al. (2004a), is a variant of the hinge loss, and is defined as follows:

$$\ell_{\text{MM}}(\mathbf{w},x_i,y_i) = \max_{y \in \mathcal{Y}} \Big[e(x_i,y_i,y) - \mathbf{w} \cdot (\mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)) \Big] \ . \tag{3}$$

2. In general the set of labels for a given example x may be a set $\mathcal{Y}(x)$ that depends on x; in fact, in our experiments on dependency parsing $\mathcal{Y}$ does depend on x. For simplicity, in this paper we use the fixed notation $\mathcal{Y}$ for all x; it is straightforward to extend our notation to the more general case.

Here $e(x_i,y_i,y)$ is some non-negative measure of the error incurred in predicting y instead of y_i as the label of x_i. We assume that $e(x_i,y_i,y_i) = 0$ for all i, so that no loss is incurred for correct prediction, and therefore $\ell_{\rm MM}(\mathbf{w},x_i,y_i)$ is always non-negative. This loss function corresponds to a maximum-margin approach, which explicitly penalizes training examples for which, for some $y \neq y_i$,

$$\mathbf{w}\cdot(\mathbf{f}(x_i,y_i)-\mathbf{f}(x_i,y)) < e(x_i,y_i,y) \ .$$

The second loss function that we will consider is based on log-linear models, and is commonly used in conditional random fields (CRFs, Lafferty et al., 2001). First define the conditional distribution

$$p(y\,|\,x;\mathbf{w}) = \frac{1}{Z_x} e^{\mathbf{w}\cdot\mathbf{f}(x,y)} \ ,$$

where $Z_x = \sum_y e^{\mathbf{w}\cdot\mathbf{f}(x,y)}$ is the partition function. The loss function is then the negative log-likelihood under the parameters $\mathbf{w}$:

$$\ell_{\rm LL}(\mathbf{w},x_i,y_i) = -\log p(y_i\,|\,x_i;\mathbf{w}) \ .$$

The function $\mathcal{L}$ is convex in $\mathbf{w}$ for both definitions $\ell_{\rm MM}$ and $\ell_{\rm LL}$. Furthermore, in both cases minimization of $\mathcal{L}$ can be re-cast as optimization of a dual convex problem. The dual problems in the two cases have a similar structure, as we describe in the next two sections.

2.2 The Log-Linear Dual

The problem of minimizing $\mathcal{L}$ with the loss function $\ell_{\rm LL}$ can be written as

$$\underline{\text{P-LL}}: \quad \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_i -\log p(y_i\,|\,x_i;\mathbf{w}) + \frac{C}{2}\|\mathbf{w}\|^2 \ .$$

This is a convex optimization problem, and has an equivalent convex dual which was derived by Lebanon and Lafferty (2002). Denote the dual variables by $u_{i,y}$ where $i = 1,\ldots,n$ and $y \in \mathcal{Y}$. We also use $\mathbf{u}$ to denote the set of all variables, and $\mathbf{u}_i$ the set of all variables corresponding to a given i. Thus $\mathbf{u} = [\mathbf{u}_1,\ldots,\mathbf{u}_n]$. We assume $\mathbf{u}$ is a column vector. Define the function $Q_{\rm LL}(\mathbf{u})$ as

$$Q_{\rm LL}(\mathbf{u}) = \sum_i \sum_y u_{i,y}\log u_{i,y} + \frac{1}{2C}\|\mathbf{w}(\mathbf{u})\|^2 \ ,$$

where

$$\mathbf{w}(\mathbf{u}) = \sum_i \sum_y u_{i,y}\mathbf{g}_{i,y} \ ,$$

and where $\mathbf{g}_{i,y} = \mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)$. We shall find the following matrix notation convenient:

$$Q_{\rm LL}(\mathbf{u}) = \sum_i \sum_y u_{i,y}\log u_{i,y} + \frac{1}{2}\mathbf{u}^T A\mathbf{u} \ , \tag{4}$$

where A is a matrix of size $n|\mathcal{Y}| \times n|\mathcal{Y}|$ indexed by pairs (i,y), and $A_{(i,y),(j,z)} = \frac{1}{C}\mathbf{g}_{i,y}\cdot\mathbf{g}_{j,z}$.

In what follows we denote the set of distributions over $\mathcal{Y}$, that is, the $|\mathcal{Y}|$-dimensional probability simplex, by Δ, as in Eq. 2. The Cartesian product of n distributions over $\mathcal{Y}$ will be denoted by Δ^n. The dual optimization problem is then

$$\underline{\text{D-LL}}: \quad \mathbf{u}^* = \begin{array}{ll} \operatorname{argmin} & Q_{\rm LL}(\mathbf{u}) \\ s.t. & \mathbf{u} \in \Delta^n \ . \end{array}$$

The minimum of D-LL is equal to -1 times the minimum of P-LL. The duality between P-LL and D-LL implies that the primal and dual solutions satisfy $C\mathbf{w}^* = \mathbf{w}(\mathbf{u}^*)$.

2.3 The Max-Margin Dual

When the loss is defined using $\ell_{\text{MM}}(\mathbf{w}, x_i, y_i)$, the primal optimization problem is as follows:

$$\underline{\text{P-MM}}: \quad \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \sum_i \max_y \Big[e(x_i, y_i, y) - \mathbf{w} \cdot (\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y)) \Big] + \frac{C}{2} \|\mathbf{w}\|^2 \ .$$

The dual of this minimization problem was derived in Taskar et al. (2004a) (see also Bartlett et al., 2005). We first define the dual objective

$$Q_{\text{MM}}(\mathbf{u}) = -\mathbf{b}^T \mathbf{u} + \frac{1}{2} \mathbf{u}^T A \mathbf{u} \ . \tag{5}$$

Here, the matrix A is as defined above and $\mathbf{b} \in \mathbb{R}^{n|\mathcal{Y}|}$ is a vector defined as $b_{i,y} = e(x_i, y_i, y)$. The convex dual for the max-margin case is then given by

$$\underline{\text{D-MM}}: \quad \mathbf{u}^* = \begin{array}{ll} \operatorname{argmin} & Q_{\text{MM}}(\mathbf{u}) \\ s.t. & \mathbf{u} \in \Delta^n \ . \end{array}$$

The minimum of D-MM is equal to -1 times the minimum of P-MM. (Note that for D-MM the minimizer $\mathbf{u}^*$ may not be unique; in this case we take $\mathbf{u}^*$ to be any member of the set of minimizers of $Q_{\text{MM}}(\mathbf{u})$). The optimal primal parameters are again related to the optimal dual parameters, through $C\mathbf{w}^* = \mathbf{w}(\mathbf{u}^*)$. Here again the constraints are that $\mathbf{u}_i$ is a distribution over $\mathcal{Y}$ for all i.

It can be seen that the D-LL and D-MM problems have a similar structure, in that they both involve minimization of a convex function $Q(\mathbf{u})$ over the set Δ^n. This will allow us to describe algorithms for both problems using a common framework.

3. Exponentiated Gradient Algorithms

In this section we describe batch and online algorithms for minimizing a convex function $Q(\mathbf{u})$ subject to the constraints $\mathbf{u} \in \Delta^n$. The algorithms can be applied to both the D-LL and D-MM optimization problems that were introduced in the previous section. The algorithms we describe are based on exponentiated gradient (EG) updates, originally introduced by Kivinen and Warmuth (1997) in the context of online learning algorithms.[3]

The EG updates rely on the following operation. Given a sequence of distributions $\mathbf{u} \in \Delta^n$, a new sequence of distributions $\mathbf{u}'$ can be obtained as

$$u'_{i,y} = \frac{1}{Z_i} u_{i,y} e^{-\eta \nabla_{i,y}} \ ,$$

where $\nabla_{i,y} = \frac{\partial Q(\mathbf{u})}{\partial u_{i,y}}$, $Z_i = \sum_{\hat{y}} u_{i,\hat{y}} e^{-\eta \nabla_{i,\hat{y}}}$ is a partition function ensuring normalization of the distribution $\mathbf{u}'_i$, and the parameter $\eta > 0$ is a learning rate. We will also use the notation $u'_{i,y} \propto u_{i,y} e^{-\eta \nabla_{i,y}}$ where the partition function should be clear from the context.

3. Kivinen and Warmuth (1997) study the online setting, as opposed to a fixed data set which we study here. They are thus not interested in minimizing a fixed objective, but rather study regret type bounds. This leads to algorithms and theoretical analyses that are different from the ones considered in the current work.

Clearly $\mathbf{u}' \in \Delta^n$ by construction. For the dual function $Q_{\text{LL}}(\mathbf{u})$ the gradient is

$$\nabla_{i,y} = 1 + \log u_{i,y} + \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{g}_{i,y} \ ,$$

and for $Q_{\text{MM}}(\mathbf{u})$ the gradient is

$$\nabla_{i,y} = -b_{i,y} + \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{g}_{i,y} \ .$$

In this paper we will consider both parallel (batch), and sequential (online) applications of the EG updates, defined as follows:

- **Batch**: At every iteration the dual variables $\mathbf{u}_i$ are simultaneously updated for all $i = 1, \ldots, n$.
- **Online**: At each iteration a single example k is chosen uniformly at random from $\{1, \ldots, n\}$ and $\mathbf{u}_k$ is updated to give $\mathbf{u}'_k$. The dual variables $\mathbf{u}_i$ for $i \neq k$ are left unchanged.

Pseudo-code for the two schemes is given in Figures 1 and 2. From here on we will refer to the batch and online EG algorithms applied to the log-linear dual as LLEG-Batch, and LLEG-Online respectively. Similarly, when applied to the max-margin dual, they will be referred to as MMEG-Batch and MMEG-Online.

Note that another plausible online algorithm would be a "deterministic" algorithm that repeatedly cycles over the training examples in a fixed order. The motivation for the alternative, randomized, algorithm is two-fold. First, we are able to prove bounds on the rate of convergence of the randomized algorithm; we have not been able to prove similar bounds for the deterministic variant. Second, our experiments show that the randomized variant converges significantly faster than the deterministic algorithm.

The EG online algorithm is essentially performing coordinate descent on the dual objective, and is similar to SVM algorithms such as SMO (Platt, 1998). For binary classification, the exact minimum of the dual objective with respect to a given coordinate can be found in closed form,[4] and more complicated algorithms such as the exponentiated-gradient method may be unnecessary. However for multi-class or structured problems, the exact minimum with respect to a coordinate $\mathbf{u}_i$ (i.e., a set of $|\mathcal{Y}|$ dual variables) cannot be found in closed form: this is a key motivation for the use of EG algorithms in this paper.

In Section 5 we give convergence proofs for the batch and online algorithms. The techniques used in the convergence proofs are quite general, and could potentially be useful in deriving EG algorithms for convex functions Q other than Q_{LL} and Q_{MM}. Before giving convergence results for the algorithms, we describe in the next section how the EG algorithms can be applied to structured problems.

4. Structured Prediction with the EG Algorithms

We now describe how the EG updates can be applied to structured prediction problems, for example parameter estimation in CRFs or natural language parsing. In structured problems the label set $\mathcal{Y}$ is typically very large, but labels can have useful internal structure. As one example, in CRFs each

4. This is true for the max-margin case. For log-linear models, minimization with respect to a single coordinate is a little more involved.

Inputs: A convex function $Q : \Delta^n \to \mathbb{R}$, a learning rate $\eta > 0$.

Initialization: Set $\mathbf{u}^1$ to a point in the interior of Δ^n.

Algorithm:

- For $t = 1, \ldots, T$, repeat:
  - For all i, y, calculate $\nabla_{i,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{i,y}}$
  - For all i, y, update $u_{i,y}^{t+1} \propto u_{i,y}^t e^{-\eta \nabla_{i,y}}$

Output: Final parameters $\mathbf{u}^{T+1}$.

Figure 1: A general batch EG Algorithm for minimizing $Q(\mathbf{u})$ subject to $\mathbf{u} \in \Delta^n$. We use $\mathbf{u}^t$ to denote the set of parameters after t iterations.

Inputs: A convex function $Q : \Delta^n \to \mathbb{R}$, a learning rate $\eta > 0$.

Initialization: Set $\mathbf{u}^1$ to a point in the interior of Δ^n.

Algorithm:

- For $t = 1, \ldots, T$, repeat:
  - Choose k_t uniformly at random from the set $\{1, 2, \ldots, n\}$
  - For all y, calculate: $\nabla_{k_t,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{k_t,y}}$
  - For all y, update $u_{k_t,y}^{t+1} \propto u_{k_t,y}^t e^{-\eta \nabla_{k_t,y}}$
  - For all $i \neq k_t$, set $\mathbf{u}_i^{t+1} = \mathbf{u}_i^t$

Output: Final parameters $\mathbf{u}^{T+1}$.

Figure 2: A general randomized online EG Algorithm for minimizing $Q(\mathbf{u})$ subject to $\mathbf{u} \in \Delta^n$.

label y is an m-dimensional vector specifying the labeling of all m vertices in a graph. In parsing each label y is an entire parse tree. In both of these cases, the number of labels typically grows exponentially quickly with respect to the size of the inputs x.

We follow the framework for structured problems described by Bartlett et al. (2005). Each label y is defined to be a set of *parts*. We use R to refer to the set of all possible parts.[5] We make the assumption that the feature vector for an entire label y decomposes into a sum over feature vectors for individual parts as follows:

$$\mathbf{f}(x, y) = \sum_{r \in y} \mathbf{f}(x, r) \ .$$

5. As with the label set $\mathcal{Y}$, the set of parts R may in general be a set $R(x)$ that depends on x. For simplicity, we assume that R is fixed.

Note that we have overloaded $\mathbf{f}$ to apply to either labels y or parts r.

As one example, consider a CRF which has an underlying graph with m nodes, and a maximum clique size of 2. Assume that each node can be labeled with one of two labels, 0 or 1. In this case the labeling of an entire graph is a vector $\mathbf{y} \in \{0,1\}^m$. Each possible input x is usually a vector in $\mathcal{X}^m$ for some set $\mathcal{X}$, although this does not have to be the case. Each part corresponds to a tuple (u,v,y_u,y_v) where (u,v) is an edge in the graph, and y_u, y_v are the labels for the two vertices u and v. The feature vector $\mathbf{f}(x,r)$ can then track any properties of the input x together with the labeled clique $r = (u,v,y_u,y_v)$. In CRFs with clique size greater than 2, each part corresponds to a labeled clique in the graph. In natural language parsing, each part can correspond to a context-free rule at a particular position in the sentence x (see Bartlett et al., 2005; Taskar et al., 2004b, for more details).

The label set $\mathcal{Y}$ can be extremely large in structured prediction problems. For example, in a CRF with an underlying graph with m nodes and k possible labels at each node, there are k^m possible labelings of the entire graph. The algorithms we have presented so far require direct manipulation of dual variables $u_{i,y}$ corresponding to each possible labeling of each training example; they will therefore be intractable in cases where there are an exponential number of possible labels. However, in this section we describe an approach that does allow an efficient implementation of the algorithms in several cases. The approach is based on the method originally described in Bartlett et al. (2005).

The key idea is as follows. Instead of manipulating the dual variables $\mathbf{u}_i$ for each i directly, we will make use of alternative data structures $\mathbf{s}_i$ for all i. Each $\mathbf{s}_i$ is a vector of real values $s_{i,r}$ for all $r \in R$. In general we will assume that there are a tractable (polynomial) number of possible parts, and therefore that the number of $s_{i,r}$ variables is also polynomial. For example, for a linear chain CRF with m nodes and k labels at every node, each part takes the form $r = (u,v,y_u,y_v)$, and there are $(m-1)k^2$ possible parts.

In the max-margin case, we follow Taskar et al. (2004a) and make the additional assumption that the error function decomposes into "local" error functions over parts:

$$e(x_i,y_i,y) = \sum_{r \in y} e(x_i,y_i,r) \ .$$

For example, when $\mathcal{Y}$ is a sequence of variables, the cost could be the Hamming distance between the correct sequence y_i and the predicted sequence y; it is straightforward to decompose the Hamming distance as a sum over parts as shown above. For brevity, in what follows we use $e_{i,r}$ instead of $e(x_i,y_i,r)$.

The $\mathbf{s}_i$ variables are used to implicitly define regular dual values $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$ where $\mathbf{p} : \mathbb{R}^{|R|} \to \Delta$ is defined as

$$p_y(\mathbf{s}) = \frac{\exp\left\{\sum_{r \in y} s_r\right\}}{\sum_{y'} \exp\left\{\sum_{r \in y'} s_r\right\}} \ .$$

To see how the $\mathbf{s}_i$ variables can be updated, consider again the EG updates on the dual $\mathbf{u}$ variables. The EG updates in all algorithms in this paper take the form

$$u'_{i,y} = \frac{u_{i,y} \exp\{-\eta \nabla_{i,y}\}}{\sum_{\hat{y}} u_{i,\hat{y}} \exp\{-\eta \nabla_{i,\hat{y}}\}} \ ,$$

where for Q_{LL}

$$\nabla_{i,y} = 1 + \log u_{i,y} + \frac{1}{C} \mathbf{w}(\mathbf{u}) \cdot (\mathbf{f}(x_i,y_i) - \mathbf{f}(x_i,y)) \ ,$$

and for Q_{MM},

$$\nabla_{i,y} = -b_{i,y} + \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot (\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y)) \ ,$$

where $b_{i,y} = e(x_i, y_i, y)$ as in Section 2.3.

Notice that, for both objective functions, the gradients can be expressed as a sum over parts. For the Q_{LL} objective function, this follows from the fact that $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$ and from the assumption that the feature vector decomposes into parts. For the Q_{MM} objective, it follows from the latter, and the assumption that the loss decomposes into parts. The following lemma describes how EG updates on the $\mathbf{u}$ variables can be restated in terms of updates to the $\mathbf{s}$ variables, provided that the gradient decomposes into parts in this way.

Lemma 1 *For a given $\mathbf{u} \in \Delta^n$, and for a given $i \in [1 \ldots n]$, take $\mathbf{u}'_i$ to be the updated value for $\mathbf{u}_i$ derived using an EG step, that is,*

$$u'_{i,y} = \frac{u_{i,y} \exp\{-\eta \nabla_{i,y}\}}{\sum_{\hat{y}} u_{i,\hat{y}} \exp\{-\eta \nabla_{i,\hat{y}}\}} \ .$$

Suppose that, for some G_i and $g_{i,r}$, we can write $\nabla_{i,y} = G_i + \sum_{r \in y} g_{i,r}$ for all y. Then if $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$ for some $\mathbf{s}_i \in \mathcal{R}^{|R|}$, and for all r we define

$$s'_{i,r} = s_{i,r} - \eta g_{i,r} \ ,$$

it follows that $\mathbf{u}'_i = \mathbf{p}(\mathbf{s}'_i)$.

Proof: We show that, for $\mathbf{u}_i = \mathbf{p}(\mathbf{s}_i)$, updating the $s_{i,r}$ as described leads to $\mathbf{p}(\mathbf{s}'_i) = \mathbf{u}'_i$. For suitable partition functions Z_i, Z'_i, and Z''_i, we can write

$$\begin{aligned}
p_y(\mathbf{s}'_i) &= \frac{\exp\left\{\sum_{r \in y}(s_{i,r} - \eta g_{i,r})\right\}}{Z_i} \\
&= \frac{u_{i,y} \exp\left\{-\eta \sum_{r \in y} g_{i,r}\right\}}{Z'_i} \\
&= \frac{u_{i,y} \exp\{-\eta(\nabla_{i,y} - G_i)\}}{Z'_i} \\
&= \frac{u_{i,y} \exp\{-\eta \nabla_{i,y}\}}{Z''_i} \\
&= u'_{i,y} \ .
\end{aligned}$$

□

In the case of the Q_{LL} objective, a suitable update is

$$s'_{i,r} = s_{i,r} - \eta \left(s_{i,r} - \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{f}(x_i, r) \right) \ .$$

In the case of the Q_{MM} objective, a suitable update is

$$s'_{i,r} = s_{i,r} - \eta \left(-e_{i,r} - \frac{1}{C}\mathbf{w}(\mathbf{u}) \cdot \mathbf{f}(x_i, r) \right) \ .$$

Inputs: Examples $\{(x_i,y_i)\}_{i=1}^n$, learning rate $\eta > 0$.

Initialization: For each $i = 1 \ldots n$, set $\mathbf{s}_i^1$ to some (possibly different) point in $\mathbb{R}^{|R|}$.

Algorithm:

- Calculate
$$\mathbf{w}^1 = \sum_i \mathbf{f}(x_i,y_i) - \sum_{i,y} p_y(\mathbf{s}_i^1)\mathbf{f}(x_i,y)$$
- For $t = 1,\ldots,T$, repeat:
  - Choose k_t uniformly at random from the set $[1,2,\ldots,n]$
  - For all $r \in R$,
$$\textbf{If optimizing } Q_{\text{LL}}\textbf{:} \quad s_{k_t,r}^{t+1} = s_{k_t,r}^t - \eta\left(s_{k_t,r}^t - \frac{1}{C}\mathbf{w}^t \cdot \mathbf{f}(x_{k_t},r)\right)$$
$$\textbf{If optimizing } Q_{\text{MM}}\textbf{:} \quad s_{k_t,r}^{t+1} = s_{k_t,r}^t - \eta\left(-e_{k_t,r} - \frac{1}{C}\mathbf{w}^t \cdot \mathbf{f}(x_{k_t},r)\right)$$
  - For all $i \neq k_t$, for all r, set $s_{i,r}^{t+1} = s_{i,r}^t$
  - Calculate
$$\begin{aligned}\mathbf{w}^{t+1} &= \sum_i \mathbf{f}(x_i,y_i) - \sum_{i,y} p_y(\mathbf{s}_i^{t+1})\mathbf{f}(x_i,y) \\ &= \mathbf{w}^t + \sum_y p_y(\mathbf{s}_{k_t}^t)\mathbf{f}(x_{k_t},y) - \sum_y p_y(\mathbf{s}_{k_t}^{t+1})\mathbf{f}(x_{k_t},y)\end{aligned}$$

Output: Final dual parameters $\mathbf{s}^{T+1}$ or primal parameters $\frac{1}{C}\mathbf{w}^{T+1}$.

Figure 3: An implementation of the algorithm in Figure 2 using a part-based representation. The algorithm uses variables $\mathbf{s}_i$ for $i = 1 \ldots n$ as a replacement for the dual variables $\mathbf{u}_i$ in Figure 2.

Because of this result, all of the EG algorithms that we have presented can be restated in terms of the **s** variables: instead of maintaining a sequence $\mathbf{u}^t = \{\mathbf{u}_1^t, \mathbf{u}_2^t, \ldots, \mathbf{u}_n^t\}$ of dual variables, a sequence $\mathbf{s}^t = \{\mathbf{s}_1^t, \mathbf{s}_2^t, \ldots, \mathbf{s}_n^t\}$ is maintained and updated using the method described in the above lemmas.[6] To illustrate this, Figure 3 gives a version of the randomized algorithm in Figure 2 that makes use of **s** variables. The batch algorithm can be implemented in a similar way.

6. Note that in the max-margin case, the optimal **u** values may have zero probabilities which correspond to infinite **s** values. This does not pose a problem, since the algorithm will indeed converge to infinite **s** values at the limit, but $\mathbf{s}^t$ will not be infinite for any finite t. For the log-linear case, the optimal **u** will never have zero values, as shown in Globerson et al. (2007).

The main computational challenge in the new algorithms comes in computing the parameter vector $\mathbf{w}(\mathbf{p}(\mathbf{s}^t))$. The value for $\mathbf{w}(\mathbf{p}(\mathbf{s}^t))$ can be expressed as a function of the *marginal probabilities* of the part variables, as follows:

$$\begin{aligned}
\mathbf{w}(\mathbf{p}(\mathbf{s}^t)) &= \sum_i \sum_y u_{i,y} \left(\mathbf{f}(x_i, y_i) - \mathbf{f}(x_i, y)\right) \\
&= \sum_i \mathbf{f}(x_i, y_i) - \sum_{i,y} p_y(\mathbf{s}_i^t) \mathbf{f}(x_i, y) \\
&= \sum_i \mathbf{f}(x_i, y_i) - \sum_{i,y} \sum_{r \in y} p_y(\mathbf{s}_i^t) \mathbf{f}(x_i, r) \\
&= \sum_i \mathbf{f}(x_i, y_i) - \sum_i \sum_{r \in R} \mu_{i,r}(\mathbf{s}_i^t) \mathbf{f}(x_i, r) \ .
\end{aligned}$$

Here the $\mu_{i,r}$ terms correspond to marginals, defined as

$$\mu_{i,r}(\mathbf{s}_i^t) = \sum_{y : r \in y} p_y(\mathbf{s}_i^t) \ .$$

The mapping from parameters $\mathbf{s}_i^t$ to marginals $\mu_{i,r}(\mathbf{s}_i^t)$ can be computed efficiently in several important cases of structured models. For example, in CRFs belief propagation can be used to efficiently calculate the marginal values, assuming that the tree-width of the underlying graph is small. In weighted context-free grammars the inside-outside algorithm can be used to calculate marginals, assuming that the set of parts R corresponds to context-free rule productions. Once marginals are computed, it is straightforward to compute $\mathbf{w}(\mathbf{p}(\mathbf{s}^t))$ and thereby implement the part-based EG algorithms.

5. Convergence Results

In this section, we provide convergence results for the EG batch and online algorithms presented in Section 3. Section 5.1 provides the key results, and the following sections give the proofs and the technical details.

5.1 Main Convergence Results

Our convergence results give bounds on how quickly the error $|Q(\mathbf{u}) - Q(\mathbf{u}^*)|$ decreases with respect to the number of iterations, T, of the algorithms. In all cases we have $|Q(\mathbf{u}) - Q(\mathbf{u}^*)| \to 0$ as $T \to \infty$.

In what follows we use $D[\mathbf{p} \| \mathbf{q}]$ to denote the KL divergence between $\mathbf{p}, \mathbf{q} \in \Delta^n$ (see Section 5.2). We also use $|A|_\infty$ to denote the maximum magnitude element of A (i.e., $|A|_\infty = \max_{(i,y),(j,z)} |A_{(i,y),(j,z)}|$). The first theorem provides results for the EG-batch algorithms, and the second for the randomized online algorithms.

Theorem 1 *For the batch algorithm in Figure 1, for Q_{LL} and Q_{MM},*

$$Q(\mathbf{u}^*) \leq Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{u}^*) + \frac{1}{\eta T} D[\mathbf{u}^* \| \mathbf{u}^1] \ , \tag{6}$$

assuming that the learning rate η satisfies $0 < \eta \leq \frac{1}{1+n|A|_\infty}$ for Q_{LL}, and $0 < \eta \leq \frac{1}{n|A|_\infty}$ for Q_{MM}. Furthermore, for Q_{LL},

$$Q(\mathbf{u}^*) \leq Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{u}^*) + \frac{e^{-\eta T}}{\eta} D[\mathbf{u}^* \| \mathbf{u}^1] \ ,$$

	Batch Algorithm	Online Algorithm
Q_{MM}	$\frac{n^2}{\varepsilon}\|A\|_\infty D[\mathbf{u}^*\|\mathbf{u}^1]$	$\frac{n}{\varepsilon}\left(\|A\|_\infty D[\mathbf{u}^*\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right)$
Q_{LL}	$n(1+n\|A\|_\infty)\log(\frac{c_1}{\varepsilon})$	$n(1+\|A\|_\infty)\log(\frac{c_2}{\varepsilon})$

Table 1: Each entry shows the amount of computation (measured in terms of the number of training sample processed using the EG updates) required to obtain $|Q(\mathbf{u}) - Q(\mathbf{u}^*)| \leq \varepsilon$ for the batch algorithm, or $\mathbf{E}\left[|Q(\mathbf{u}) - Q(\mathbf{u}^*)|\right] \leq \varepsilon$ for the online algorithm, for a given $\varepsilon > 0$. The constants are $c_1 = (1+n|A|_\infty)D[\mathbf{u}^*\|\mathbf{u}^1]$, and $c_2 = \left[(1+|A|_\infty)D[\mathbf{u}^*\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*))\right]$.

assuming again that $0 < \eta \leq \frac{1}{1+n|A|_\infty}$.

The randomized online algorithm will produce different results at every run, since different points will be processed on different runs. Our main result for this algorithm characterizes the mean value of the objective $Q(\mathbf{u}^{T+1})$ when averaged over all possible random orderings of points. The result implies that this mean will converge to the optimal value $Q(\mathbf{u}^*)$.

Theorem 2 *For the randomized algorithm in Figure 2, for* Q_{LL} *and* Q_{MM}*,*

$$Q(\mathbf{u}^*) \leq \mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{u}^*) + \frac{n}{\eta T} D[\mathbf{u}^*\|\mathbf{u}^1] + \frac{n}{T}\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] , \tag{7}$$

assuming that the learning rate η *satisfies* $0 < \eta \leq \frac{1}{1+|A|_\infty}$ *for* Q_{LL}*, and* $0 < \eta \leq \frac{1}{|A|_\infty}$ *for* Q_{MM}*. Furthermore, for* Q_{LL}*, for the algorithm in Figure 2,*

$$Q(\mathbf{u}^*) \leq \mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{u}^*) + e^{-\frac{\eta T}{n}}\left[\frac{1}{\eta} D[\mathbf{u}^*\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,$$

assuming again that $0 < \eta \leq \frac{1}{1+|A|_\infty}$.

The above result characterizes the average behavior of the randomized algorithm, but does not provide guarantees for any specific run of the algorithm. However, by applying the standard approach of repeated sampling (see, for example, Mitzenmacher and Upfal, 2005; Shalev-Shwartz et al., 2007), one can obtain a solution that, with high probability, does not deviate by much from the average behavior. In what follows, we briefly outline this derivation.

Note that the random variable $Q(\mathbf{u}^{T+1}) - Q(\mathbf{u}^*)$ is nonnegative, and so by Markov's inequality, it satisfies

$$\Pr\left\{Q(\mathbf{u}^{T+1}) - Q(\mathbf{u}^*) \geq 2\left(\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] - Q(\mathbf{u}^*)\right)\right\} \leq \frac{1}{2} .$$

Given some $\delta > 0$, if we run the algorithm $k = \log_2(\frac{1}{\delta})$ times,[7] each time with T iterations, and choose the best $\hat{\mathbf{u}}$ of these k results, we see that

$$\Pr\left\{Q(\hat{\mathbf{u}}) - Q(\mathbf{u}^*) \geq 2\left(\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] - Q(\mathbf{u}^*)\right)\right\} \leq \delta .$$

7. Assume for simplicity that $\log_2(\frac{1}{\delta})$ is integral.

Thus, for any desired confidence $1-\delta$, we can obtain a solution that is within a factor of 2 of the bound for T iterations in Theorem 2 by using $T\log_2(\frac{1}{\delta})$ iterations. In our experiments, we found that repeated trials of the randomized algorithm did not yield significantly different results.

The first consequence of the two theorems above is that the batch and randomized online algorithms converge to a $\mathbf{u}$ with the optimal value $Q(\mathbf{u}^*)$. This follows since Equations 6 and 7 imply that as $T \to \infty$ the value of $Q(\mathbf{u}^{T+1})$ approaches $Q(\mathbf{u}^*)$.

The second consequence is that for a given $\varepsilon > 0$ we can find the number of iterations needed to reach a $\mathbf{u}$ such that $|Q(\mathbf{u}) - Q(\mathbf{u}^*)| \leq \varepsilon$ for the batch algorithm or $\mathbf{E}\left[|Q(\mathbf{u}) - Q(\mathbf{u}^*)|\right] \leq \varepsilon$ for the online algorithm. Table 1 shows the computation required by the different algorithms, where the computation is measured by the number of training examples that need to be processed using the EG updates.[8] The entries in the table assume that the maximum possible learning rates are used for each of the algorithms—that is, $\frac{1}{1+n|A|_\infty}$ for LLEG-Batch, $\frac{1}{1+|A|_\infty}$ for LLEG-Online, $\frac{1}{n|A|_\infty}$ for MMEG-batch, and $\frac{1}{|A|_\infty}$ for MMEG-Online.

Crucially, note that these rates suggest that the online algorithms are significantly more efficient than the batch algorithms; specifically, the bounds suggest that the online algorithms require a factor of n less computation in both the Q_{LL} and Q_{MM} cases. Thus these results suggest that the randomized online algorithm should converge much faster than the batch algorithm. Roughly speaking, this is a direct consequence of the learning rate η being a factor of n larger in the online case (see also Section 9). This prediction is confirmed in our empirical evaluations, which show that the online algorithm is far more efficient than the batch algorithm.

A second important point is that the rates for Q_{LL} lead to an $O(\log(\frac{1}{\varepsilon}))$ dependence on the desired accuracy ε, which is a significant improvement over Q_{MM}, which has an $O(\frac{1}{\varepsilon})$ dependence. Note that the $O(\frac{1}{\varepsilon})$ dependence for Q_{MM} is similar to several other max-margin algorithms in the literature (see Section 6 for more discussion).

To gain further intuition into the order of magnitude of iterations required, note that the factor $D[\mathbf{u}^*\|\mathbf{u}^1]$ which appears in the above expressions is at most $n\log|\mathcal{Y}|$, which can be achieved by setting $\mathbf{u}_i^1$ to be the uniform distribution over $\mathcal{Y}$ for all i. Also, the value of $|A|_\infty$ can easily be seen to be $\frac{1}{C}\max_{i,y}\|\mathbf{g}_{i,y}\|^2$.

In the remainder of this section we give proofs of the results in Theorems 1 and 2. In doing so, we also give theorems that apply to the optimization of general convex functions $Q : \Delta^n \to \mathbb{R}$.

5.2 Divergence Measures

Before providing convergence proofs, we define several divergence measures between distributions. Define the KL divergence between two distributions $\mathbf{u}_i, \mathbf{v}_i \in \Delta$ to be

$$D[\mathbf{u}_i\|\mathbf{v}_i] = \sum_y u_{i,y}\log\frac{u_{i,y}}{v_{i,y}} \ .$$

Given two sets of n distributions $\mathbf{u}, \mathbf{v} \in \Delta^n$ define their KL divergence as

$$D[\mathbf{u}\|\mathbf{v}] = \sum_i D[\mathbf{u}_i\|\mathbf{v}_i] \ .$$

8. Note that if we run the batch algorithm for T iterations (as in the figure), nT training examples are processed. In contrast, running the online algorithm for T iterations (again, as shown in the figure) only requires T training examples to be processed. It is important to take this into account when comparing the rates in Theorems 1 and 2; this is the motivation for measuring computation in terms of the number of examples that are processed.

Next, we consider a more general class of divergence measures, Bregman divergences (e.g., see Bregman, 1967; Censor and Zenios, 1997; Kivinen and Warmuth, 1997). Given a convex function $Q(\mathbf{u})$, the Bregman divergence between $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$B_Q[\mathbf{u}\|\mathbf{v}] = Q(\mathbf{u}) - Q(\mathbf{v}) - \nabla Q(\mathbf{v}) \cdot (\mathbf{u} - \mathbf{v}) \ .$$

Convexity of Q implies $B_Q[\mathbf{u}\|\mathbf{v}] \geq 0$ for all $\mathbf{u}, \mathbf{v} \in \Delta^n$.

Note that the Bregman divergence with $Q(\mathbf{u}) = \sum_{i,y} u_{i,y} \log u_{i,y}$ is the KL divergence. We shall also be interested in the Mahalanobis distance

$$M_A[\mathbf{u}\|\mathbf{v}] = \frac{1}{2}(\mathbf{u} - \mathbf{v})^T A (\mathbf{u} - \mathbf{v}) \ ,$$

which is the Bregman divergence for $Q(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T A \mathbf{u}$.

In what follows, we also use the L_p norm defined for $\mathbf{x} \in \mathbb{R}^m$ as $\|\mathbf{x}\|_p = \sqrt[p]{\sum_i |x_i|^p}$.

5.3 Dual Improvement and Bregman Divergence

In this section we provide a useful lemma that determines when the EG updates in the batch algorithm will result in monotone improvement of $Q(\mathbf{u})$. The lemma requires a condition on the relation between the Bregman and KL divergences which we define as follows (the second part of the definition will be used in the next section):

Definition 5.1 : A convex function $Q : \Delta^n \to \mathbb{R}$ is τ-upper-bounded for some $\tau > 0$ if for any $\mathbf{p}, \mathbf{q} \in \Delta^n$,

$$B_Q[\mathbf{p}\|\mathbf{q}] \leq \tau D[\mathbf{p}\|\mathbf{q}] \ .$$

In addition, we say $Q(\mathbf{u})$ is (μ, τ)-bounded for constants $0 < \mu < \tau$ if for any $\mathbf{p}, \mathbf{q} \in \Delta^n$,

$$\mu D[\mathbf{p}\|\mathbf{q}] \leq B_Q[\mathbf{p}\|\mathbf{q}] \leq \tau D[\mathbf{p}\|\mathbf{q}] \ .$$

The next lemma states that if $Q(\mathbf{u})$ is τ-upper-bounded, then the change in the objective as a result of an EG update can be related to the KL divergence between consecutive values of the dual variables.

Lemma 2 *If $Q(\mathbf{u})$ is τ-upper-bounded, then if η is chosen such that $0 < \eta \leq \frac{1}{\tau}$, it holds that for all t in the batch algorithm (Figure 1):*

$$Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \ .$$

Proof: Given a $\mathbf{u}^t$, the EG update is

$$u^{t+1}_{i,y} = \frac{1}{Z^t_i} u^t_{i,y} e^{-\eta \nabla^t_{i,y}} \ ,$$

where

$$\nabla^t_{i,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{i,y}} \ , \qquad Z^t_i = \sum_{\hat{y}} u^t_{i,\hat{y}} e^{-\eta \nabla^t_{i,y}} \ .$$

Simple algebra yields

$$\sum_i \left(D[\mathbf{u}_i^t \| \mathbf{u}_i^{t+1}] + D[\mathbf{u}_i^{t+1} \| \mathbf{u}_i^t] \right) = \eta \sum_{i,y} (u_{i,y}^t - u_{i,y}^{t+1}) \nabla_{i,y}^t \ .$$

Equivalently, using the notation for KL divergence between multiple distributions:

$$D[\mathbf{u}^t \| \mathbf{u}^{t+1}] + D[\mathbf{u}^{t+1} \| \mathbf{u}^t] = \eta (\mathbf{u}^t - \mathbf{u}^{t+1}) \cdot \nabla Q(\mathbf{u}^t) \ .$$

The definition of the Bregman divergence B_Q then implies

$$-\eta B_Q[\mathbf{u}^{t+1} \| \mathbf{u}^t] + D[\mathbf{u}^t \| \mathbf{u}^{t+1}] + D[\mathbf{u}^{t+1} \| \mathbf{u}^t] = \eta (Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1})) \ . \tag{8}$$

Since $Q(\mathbf{u})$ is τ-upper-bounded and $\eta \leq \frac{1}{\tau}$ it follows that $D[\mathbf{u}^{t+1} \| \mathbf{u}^t] \geq \eta B_Q[\mathbf{u}^{t+1} \| \mathbf{u}^t]$, and together with Eq. 8 we obtain the desired result $\eta(Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1})) \geq D[\mathbf{u}^t \| \mathbf{u}^{t+1}]$. □

Note that the condition in the lemma may be weakened to requiring that $\tau D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \geq B_Q[\mathbf{u}^t \| \mathbf{u}^{t+1}]$ for all t. For simplicity, we require the condition for all $\mathbf{p}, \mathbf{q} \in \Delta^n$. Note also that $D[\mathbf{p} \| \mathbf{q}] \geq 0$ for all $\mathbf{p}, \mathbf{q} \in \Delta^n$, so the lemma implies that for an appropriately chosen η, the EG updates always decrease the objective $Q(\mathbf{u})$. We next show that the log-linear dual $Q_{\text{LL}}(\mathbf{u})$ is in fact τ-upper-bounded.

Lemma 3 *Define $|A|_\infty$ to be the maximum magnitude of any element of A, that is, $|A|_\infty = \max_{(i,y),(j,z)} |A_{(i,y),(j,z)}|$. Then $Q_{\text{LL}}(\mathbf{u})$ is τ_{LL}-upper-bounded with $\tau_{\text{LL}} = 1 + n|A|_\infty$.*

Proof: First notice that the Bregman divergence B_Q is linear in Q. Thus, for any $\mathbf{p}, \mathbf{q} \in \Delta^n$ we can write $B_{Q_{\text{LL}}}$ as a sum of two terms (see Eq. 4).

$$B_{Q_{\text{LL}}}[\mathbf{p} \| \mathbf{q}] = D[\mathbf{p} \| \mathbf{q}] + M_A[\mathbf{p} \| \mathbf{q}] \ .$$

We first bound $M_A[\mathbf{p} \| \mathbf{q}]$ in terms of squared L_1 distance between $\mathbf{p}$ and $\mathbf{q}$. Denote $\mathbf{r} = \mathbf{p} - \mathbf{q}$. Then:

$$M_A[\mathbf{p} \| \mathbf{q}] = \frac{1}{2} \sum_{i,y,j,z} r_{i,y} r_{j,z} A_{(i,y),(j,z)} \leq \frac{|A|_\infty}{2} \sum_{i,y,j,z} |r_{i,y}| |r_{j,z}| = \frac{|A|_\infty}{2} \|\mathbf{p} - \mathbf{q}\|_1^2 \ .$$

Next, we use the inequality $D[p_1 \| p_2] \geq \frac{1}{2} \|p_1 - p_2\|_1^2$ (also known as Pinsker's inequality, see Cover and Thomas, 1991, p. 300), which holds for any two distributions p_1 and p_2. Consider the two distributions $\hat{\mathbf{p}} = \frac{1}{n}\mathbf{p}$ and $\hat{\mathbf{q}} = \frac{1}{n}\mathbf{q}$, each defined over an alphabet of size $n|\mathcal{Y}|$. Then it follows that:[9]

$$\frac{|A|_\infty}{2} \|\mathbf{p} - \mathbf{q}\|_1^2 = \frac{n^2 |A|_\infty}{2} \|\hat{\mathbf{p}} - \hat{\mathbf{q}}\|_1^2 \leq n^2 |A|_\infty D[\hat{\mathbf{p}} \| \hat{\mathbf{q}}] = n|A|_\infty D[\mathbf{p} \| \mathbf{q}] \ ,$$

and thus $M_A[\mathbf{p} \| \mathbf{q}] \leq n|A|_\infty D[\mathbf{p} \| \mathbf{q}]$. So for the Bregman divergence of $Q_{\text{LL}}(\mathbf{u})$ we obtain

$$B_{Q_{\text{LL}}}[\mathbf{p} \| \mathbf{q}] \leq (1 + n|A|_\infty) D[\mathbf{p} \| \mathbf{q}] \ ,$$

yielding the desired result. □

The next lemma shows that a similar result can be obtained for the Q_{MM} objective.

9. Note that $D[\hat{\mathbf{p}} \| \hat{\mathbf{q}}]$ is a divergence between two distributions over $|\mathcal{Y}|n$ symbols and $D[\mathbf{p} \| \mathbf{q}]$ is a divergence between two sets of n distributions over $|\mathcal{Y}|$ symbols.

Lemma 4 *The function $Q_{\text{MM}}(\mathbf{u})$ is τ_{MM}-upper-bounded with $\tau_{\text{MM}} = n|A|_\infty$.*

Proof: For Q_{MM} defined in Eq. 5, we have

$$B_{Q_{\text{MM}}}[\mathbf{p}\|\mathbf{q}] = M_A[\mathbf{p}\|\mathbf{q}] \ .$$

We can then use a similar derivation to that of Lemma 3 to obtain the result. □

We thus have that the condition in Lemma 2 is satisfied for both the $Q_{\text{LL}}(\mathbf{u})$ and $Q_{\text{MM}}(\mathbf{u})$ objectives, implying that their EG updates result in monotone improvement of the objective, for a suitably chosen η:

Corollary 1 *The LLEG-Batch algorithm with $0 < \eta \leq \frac{1}{\tau_{\text{LL}}}$ satisfies for all t*

$$Q_{\text{LL}}(\mathbf{u}^t) - Q_{\text{LL}}(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \ ,$$

and the MMEG-Batch algorithm with $0 < \eta \leq \frac{1}{\tau_{\text{MM}}}$ satisfies for all t

$$Q_{\text{MM}}(\mathbf{u}^t) - Q_{\text{MM}}(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t \| \mathbf{u}^{t+1}] \ .$$

5.4 Convergence Rates for the EG Batch Algorithms

The previous section showed that for appropriate choices of the learning rate η, the batch EG updates are guaranteed to improve the Q_{LL} and Q_{MM} loss functions at each iteration. In this section we build directly on these results, and address the following question: how many iterations does the batch EG algorithm require so that the $|Q(\mathbf{u}^t) - Q(\mathbf{u})| \leq \varepsilon$ for a given $\varepsilon > 0$? We show that as long as $Q(\mathbf{u})$ is τ-upper-bounded, the number of iterations required is $O(\frac{1}{\varepsilon})$. This bound thus holds for both the log-linear and max-margin batch algorithms. Next, we show that if $Q(\mathbf{u})$ is (μ, τ)-bounded, the rate can be significantly improved to requiring $O(\log(\frac{1}{\varepsilon}))$ iterations. We conclude by showing that $Q_{\text{LL}}(\mathbf{u})$ is (μ, τ)-bounded, implying that the $O(\log(\frac{1}{\varepsilon}))$ rate holds for LLEG-Batch.

The following result gives an $O(\frac{1}{\varepsilon})$ rate for Q_{LL} and Q_{MM}:

Lemma 5 *If $Q(\mathbf{u})$ is τ-upper-bounded and $0 \leq \eta \leq \frac{1}{\tau}$, then after T iterations of the EG-Batch algorithm, for any $\mathbf{z} \in \Delta^n$ including $\mathbf{z} = \mathbf{u}^*$,*

$$Q(\mathbf{u}^{T+1}) - Q(\mathbf{z}) \leq \frac{1}{\eta T} D[\mathbf{z}\|\mathbf{u}^1] \ .$$

Proof: See Appendix A. □

The lemma implies that to get ε-close to the optimal objective value, $O(\frac{1}{\varepsilon})$ iterations are required—more precisely, if a choice of $\eta = \frac{1}{\tau}$ is made, then at most $\frac{\tau}{\varepsilon} D[\mathbf{z}\|\mathbf{u}^1]$ iterations are required. Since its conditions are satisfied by both $Q_{\text{LL}}(\mathbf{u})$ and $Q_{\text{MM}}(\mathbf{u})$ (given an appropriate choice of η) the result holds for both the LLEG-Batch and MMEG-Batch algorithms.

A much improved rate may be obtained if $Q(\mathbf{u})$ is not only τ-upper-bounded, but also (μ, τ)-bounded (see Definition 5.1).

Lemma 6 *If $Q(\mathbf{u})$ is (μ,τ)-bounded and $0 < \eta \leq \frac{1}{\tau}$ then after T iterations of the EG-Batch algorithm, for any $\mathbf{z} \in \Delta^n$ including $\mathbf{z} = \mathbf{u}^*$,*

$$Q(\mathbf{u}^{T+1}) - Q(\mathbf{z}) \leq \frac{e^{-\eta\mu T}}{\eta} D[\mathbf{z}||\mathbf{u}^1] \ .$$

Proof: See Appendix B. □

The lemma implies that an accuracy of ε may be achieved by using $O(\log(\frac{1}{\varepsilon}))$ iterations.

To see why $Q_{\text{LL}}(\mathbf{u})$ is (μ,τ)-bounded note that for any $\mathbf{p},\mathbf{q} \in \Delta^n$,

$$B_{Q_{\text{LL}}}[\mathbf{p}\|\mathbf{q}] = D[\mathbf{p}\|\mathbf{q}] + M_A[\mathbf{p}\|\mathbf{q}] \geq D[\mathbf{p}\|\mathbf{q}] \ ,$$

implying (together with Lemma 3) that $Q_{\text{LL}}(\mathbf{u})$ is $(1,\tau_{\text{LL}})$-bounded.

Finally, note that Lemmas 5 and 6, together with the facts that Q_{LL} is $(1,\tau_{\text{LL}})$-bounded and Q_{MM} is τ_{MM}-upper-bounded, imply Theorem 1 of Section 5.1.

5.5 Convergence Results for the Randomized Online Algorithm

This section analyzes the rate of convergence of the randomized online algorithm in Figure 2. Before stating the results, we need some definitions. We will use $Q_{\mathbf{u},i} : \Delta \rightarrow \mathbb{R}$ to be the function defined as

$$Q_{\mathbf{u},i}(\mathbf{v}) = Q(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{i-1}, \mathbf{v}, \mathbf{u}_{i+1}, \ldots, \mathbf{u}_n) \ ,$$

for any $\mathbf{v} \in \Delta$. We denote the Bregman divergence associated with $Q_{\mathbf{u},i}$ as $B_{Q_{\mathbf{u},i}}[\mathbf{x}\|\mathbf{y}]$. We then introduce the following definitions:

Definition 5.2: *A convex function $Q : \Delta^n \rightarrow \mathbb{R}$ is τ-online-upper-bounded for some $\tau > 0$ if for any $i \in 1 \ldots n$ and for any $\mathbf{p},\mathbf{q} \in \Delta$,*

$$B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] \leq \tau D[\mathbf{p}\|\mathbf{q}] \ .$$

In addition, Q is (μ,τ)-online-bounded for $0 < \mu < \tau$ if Q is τ-online-upper-bounded, and in addition, for any $\mathbf{p},\mathbf{q} \in \Delta^n$,

$$\mu D[\mathbf{p}\|\mathbf{q}] \leq B_Q[\mathbf{p}\|\mathbf{q}] \ .$$

Note that the lower bound in the above definition refers to B_Q and not to $B_{Q_{\mathbf{u},i}}$. Also, note that if a function is (μ,τ)-online-bounded then it must also be τ-online-upper-bounded.

The following lemma then gives results for the Q_{LL} and Q_{MM} functions:

Lemma 7 *The log-linear dual $Q_{\text{LL}}(\mathbf{u})$ is (μ,τ)-online-bounded for $\mu = 1$ and $\tau = 1 + |A|_\infty$. The max-margin dual $Q_{\text{MM}}(\mathbf{u})$ is τ-online-upper-bounded for $\tau = |A|_\infty$.*

Proof: See Appendix C. □

For any τ-online-upper-bounded Q, the online EG algorithm converges at an $O(\frac{1}{\varepsilon})$ rate, as shown by the following lemma.

Lemma 8 *Consider the algorithm in Figure 2 applied to a convex function $Q(\mathbf{u})$ that is τ-online-upper-bounded. If $\eta > 0$ is chosen such that $\eta \leq \frac{1}{\tau}$, then it follows that for all $\mathbf{z} \in \Delta^n$*

$$\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{z}) + \frac{n}{\eta T} D[\mathbf{z}\|\mathbf{u}^1] + \frac{n}{T}\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,$$

where $\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right]$ is the expected value of $Q(\mathbf{u}^{T+1})$, and $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}\in\Delta^n} Q(\mathbf{u})$.

Proof: See Appendix D. □

The previous lemma shows, in particular, that the online EG algorithm converges at an $O(\frac{1}{\varepsilon})$ rate for the function Q_{MM}. However, we can prove a faster rate of convergence for Q_{LL}, which is (μ,τ)-online-bounded. The following lemma shows that such functions exhibit an $O(\log(\frac{1}{\varepsilon}))$ rate of convergence.

Lemma 9 *Consider the algorithm in Figure 2 applied to a convex function $Q(\mathbf{u})$ that is (μ,τ)-online-bounded. If $\eta > 0$ is chosen such that $\eta \leq \frac{1}{\tau}$, then it follows that for all $\mathbf{z} \in \Delta^n$*

$$\mathbf{E}\left[Q(\mathbf{u}^{T+1})\right] \leq Q(\mathbf{z}) + e^{-\frac{\eta\mu T}{n}}\left[\frac{1}{\eta} D[\mathbf{z}\|\mathbf{u}^1] + Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,$$

where $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}\in\Delta^n} Q(\mathbf{u})$.

Proof: See Appendix E. □

Note that Lemmas 7, 9 and 8 complete the proof of Theorem 2 in Section 5.1.

6. Related Work

The idea of solving regularized loss-minimization problems via their convex duals has been addressed in several previous papers. Here we review those, specifically focusing on the log-linear and max-margin problems.

Zhang (2002) presented a study of convex duals of general regularized loss functions, and provided a methodology for deriving such duals. He also considered a general procedure for solving such duals by optimizing one coordinate at a time. However, it is not clear how to implement this procedure in the structured learning case (i.e., when $|\mathcal{Y}|$ is large), and convergence rates are not given.

In the specific context of log-linear models, several papers have addressed dual optimization. Earlier work (Jaakkola and Haussler, 1999; Keerthi et al., 2005; Zhu and Hastie, 2001) treated the logistic regression model, a simpler version of a CRF. In the binary logistic regression case, there is essentially one parameter u_i per example with the constraint $0 \leq u_i \leq 1$, and therefore simple line-search methods can be used for optimization. Minka (2003) presents empirical results which show that this approach performs similarly to conjugate gradient. The problem becomes much harder when $\mathbf{u}_i$ is constrained to be a distribution over many labels, as in the case discussed here. Recently, Memisevic (2006) addressed this setting, and suggests optimizing $\mathbf{u}_i$ by transferring probability mass between two labels y_1, y_2 while keeping the distribution normalized. This requires a strategy for choosing these two labels, and the author suggests one which seems to perform well.

While some previous work on log-linear models proved convergence of dual methods (e.g., Keerthi et al., 2005), we are not aware of rates of convergence that have been reported in this context. Convergence rates for related algorithms, in particular a generalization of EG, known as the Mirror-Descent algorithm, have been studied in a more general context in the optimization literature. For instance, Beck and Teboulle (2003) describe convergence results which apply to quite general definitions of $Q(\mathbf{u})$, but which have only $O(\frac{1}{\varepsilon^2})$ convergence rates, as compared to our results of $O(\frac{1}{\varepsilon})$ and $O(\log(\frac{1}{\varepsilon}))$ for the max-margin and log-linear cases respectively. Also, their work considers optimization over a single simplex, and does not consider online-like algorithms such as the one we have presented.

For max-margin models, numerous dual methods have been suggested, an earlier example being the SMO algorithm of Platt (1998). Such methods optimize subsets of the $\mathbf{u}$ parameters in the dual SVM formulation (see also Crammer and Singer, 2002). Analysis of a similar algorithm (Hush et al., 2006) results in an $O(\frac{1}{\varepsilon})$ rate, similar to the one we have here. Another algorithm for solving SVMs via the dual is the multiplicative update method of Sha et al. (2007). These updates are shown to converge to the optimum of the SVM dual, but convergence rate has not been analyzed, and extension to the structured case seems non-trivial. An application of EG to binary SVMs was previously studied by Cristianini et al. (1998). They show convergence rates of $O(\frac{1}{\varepsilon^2})$, that are slower than our $O(\frac{1}{\varepsilon})$, and no extension to structured learning (or multi-class) is discussed.

Recently, several new algorithms have been presented, along with a rate of convergence analysis (Joachims, 2006; Shalev-Shwartz et al., 2007; Teo et al., 2007; Tsochantaridis et al., 2004; Taskar et al., 2006). All of these algorithms are similar to ours in having a relatively low dependence on n in terms of memory and computation. Among these, Shalev-Shwartz et al. (2007), Teo et al. (2007) and Taskar et al. (2006) present an $O(\frac{1}{\varepsilon})$ rate, but where accuracy is measured in the primal or via the duality gap, and not in the dual as in our analysis. Thus, it seems that a rate of $O(\frac{1}{\varepsilon})$ is currently the best known result for algorithms that have a relatively low dependence on n (general QP solvers, which may have $O(\log(\frac{1}{\varepsilon}))$ behavior, generally have a larger dependence on n, both in time and space). Note that, as in our analysis, all these convergence rates depend on $|A|_\infty$.

Finally, we emphasize again that the EG algorithm is substantially different from stochastic gradient and stochastic subgradient approaches (LeCun et al., 1998; Nedic and Bertsekas, 2001; Shalev-Shwartz et al., 2007; Vishwanathan et al., 2006). EG and stochastic gradient methods are similar in that they both process a single training example at a time. However, EG corresponds to block-coordinate descent in the dual, and uses the *exact* gradient with respect to the block being updated. In contrast, stochastic gradient methods directly optimize the primal problem, and at each update use a single example to *approximate* the gradient (or subgradient) of the primal objective function.

7. Experiments on Regularized Log-Likelihood

In this section we analyze the performance of the EG algorithms for optimization of regularized log-likelihood. We describe experiments on two tasks: first, the MNIST digit classification task, which is a multiclass classification task; second, a log-linear model for a structured natural-language dependency-parsing task. In each case we first give results for the EG method, and then compare

its performance to L-BFGS (Byrd et al., 1995), which is a batch gradient descent method, and to stochastic gradient descent.[10]

We do not report results on LLEG-Batch, since we found it to converge much more slowly than the online algorithm. This is expected from our theoretical results, which anticipate a factor of n speed-up for the online algorithm. We also report experiments comparing the randomized online algorithm to a deterministic online EG algorithm, where samples are drawn in a fixed order (e.g., the algorithm first visits the first example, then the second, etc.).

Although EG is guaranteed to converge for an appropriately chosen η, it turns out to be beneficial to use an adaptive learning rate. Here we use the following simple strategy: we first consider only 10% of the data-set, and find a value of η that results in monotone improvement for at least 95% of the samples. Denote this value by η^{ini} (for the experiments in Section 7.1 we simply use $\eta^{\text{ini}} = 0.5$). For learning over the entire data-set, we keep a learning rate η_i for each sample i (where $i = 1,\ldots,n$), and initialize this rate to η^{ini} for all points. When sample i is visited, we halve η_i until an improvement in the objective is obtained. Finally, after the update, we multiply η_i by 1.05, so that it does not decrease monotonically.

It is important that when updating a single example using the online algorithms, the improvement (or decrease) in the dual can be easily evaluated, allowing the halving strategy described in the previous paragraph to be implemented efficiently. If the current dual parameters are $\mathbf{u}$, the i'th coordinate is selected, and the EG updates then map $\mathbf{u}_i$ to $\mathbf{u}'_i$, the change in the dual objective is

$$\sum_y u'_{i,y} \log u'_{i,y} + \frac{1}{2C} \left\| \mathbf{w}(\mathbf{u}) + \sum_y \left(u'_{i,y} - u_{i,y}\right) \mathbf{g}_{i,y} \right\|^2 - \sum_y u_{i,y} \log u_{i,y} - \frac{1}{2C} \|\mathbf{w}(\mathbf{u})\|^2 \ .$$

The primal parameters $\mathbf{w}(\mathbf{u})$ are maintained throughout the algorithm (see Figure 3), so that this change in the dual objective can be calculated efficiently. A similar method can be used to calculate the change in the dual objective in the max-margin case.

We measure the performance of each training algorithm (the EG algorithms, as well as the batch gradient and stochastic gradient methods) as a function of the amount of computation spent. Specifically, we measure computation in terms of the number of times each training example is visited. For EG, an example is considered to be visited for every value of η that is tested on it. For L-BFGS, all examples are visited for every evaluation performed by the line-search routine. We define the measure of *effective iterations* to be the number of examples visited, divided by n. In the following sections we compare the algorithms in terms of their performance as a function of the effective number of iterations. A comparison in terms of running time is provided in Appendix F; there is little difference between the timed comparisons and the results presented in this section.

7.1 Multi-class Classification

We first conducted multi-class classification experiments on the MNIST classification task. Examples in this data set are images of handwritten digits represented as 784-dimensional vectors. We used a training set of 59k examples, and a validation set of 10k examples.[11] Note that since we

10. We also experimented with conjugate gradient algorithms, but since these resulted in worse performance than L-BFGS, we do not report these results here.

11. In reporting results, we consider only *validation* error; that is, error computed during the training process on a validation set. This measure is often used in early-stopping of algorithms, and is therefore of interest in the current context. We do not report test error since our main focus is algorithmic.

only use a linear kernel, accuracy results are not competitive with state of the art classifiers which use non-linear kernels (e.g., see Cortes and Vapnik, 1995). We define a ten-class logistic-regression model where

$$p(y|x) \propto e^{\mathbf{x}\cdot\mathbf{w}_y} ,$$

and $\mathbf{x}, \mathbf{w}_y \in \mathbb{R}^{784}, y \in \{1, \ldots, 10\}$.

Models were trained for various values of the regularization parameter C: specifically, we tried values of C equal to 1000, 100, 10, 1, 0.1, and 0.01. Convergence of the EG algorithm for low values of C (i.e., 0.1 and 0.01) was found to be slow; we discuss this issue more in Section 7.1.1, arguing that it is not a serious problem.

Figure 4 shows plots of the validation error versus computation for C equal to 1000, 100, 10, and 1, when using the EG algorithm. For C equal to 10 or more, convergence is fast. For $C = 1$ convergence is somewhat slower. Note that there is little to choose between $C = 10$ and $C = 1$ in terms of validation error.

Figure 5 shows plots of the primal and dual objective functions for different values of C. To obtain the primal objective values, we used the EG weight vector $\frac{1}{C}\mathbf{w}(\mathbf{u}^t)$. Note that EG does not explicitly minimize the primal objective function, so the EG primal will not necessarily decrease at every iteration. Nevertheless, our experiments show that the EG primal decreases quite quickly. Figure 6 shows how the duality gap decreases with the amount of computation spent (the duality gap is the difference between the primal and dual values at each iteration). The log of the duality gap decreases more-or-less linearly with the amount of computation spent, as predicted by the $O(\log(\frac{1}{\varepsilon}))$ bounds on the rate of convergence.[12]

Finally, we compare the deterministic and randomized versions of the EG algorithm. Figure 7 shows the primal and dual objectives for both algorithms. It can be seen that the randomized algorithm is clearly much faster to converge. This is even more evident when plotting the duality gap, which converges much faster to zero in the case of the randomized algorithm. These results give empirical evidence that the randomized strategy is to be preferred over a fixed ordering of the training examples (note that we have been able to prove bounds on convergence rate for the randomized algorithm, but have not been able to prove similar bounds for the deterministic case).

7.1.1 CONVERGENCE FOR LOW VALUES OF C

As mentioned in the previous section, convergence of the EG algorithm for low values of C can be very slow. This is to be expected from the bounds on convergence, which predict that convergence time should scale linearly with $\frac{1}{C}$ (other algorithms, e.g., see Shalev-Shwartz et al., 2007, also require $O(\frac{1}{C})$ time for convergence). This is however, not a serious problem on the MNIST data, where validation error has reached a minimum point for around $C = 10$ or $C = 1$.

If convergence for small values of C is required, one strategy we have found effective is to start C at a higher value, then "anneal" it towards the target value. For example, see Figure 8 for results for $C = 1$ using one such annealing scheme. For this experiment, if we take t to be the number of iterations over the training set, where for any t we have processed $t \times n$ training examples, we set $C = 10$ for $t \leq 5$, and set $C = 1 + 9 \times 0.7^{t-5}$ for $t > 5$. Thus C starts at 10, then decays exponentially quickly towards the target value of 1. It can be seen that convergence is significantly faster for the annealed method. The intuition behind this method is that the solution to the dual problem for

12. The rate results presented in this paper are for dual accuracy, but it is straightforward to obtain an $O(\log(\frac{1}{\varepsilon}))$ for the duality gap in the log-linear case.

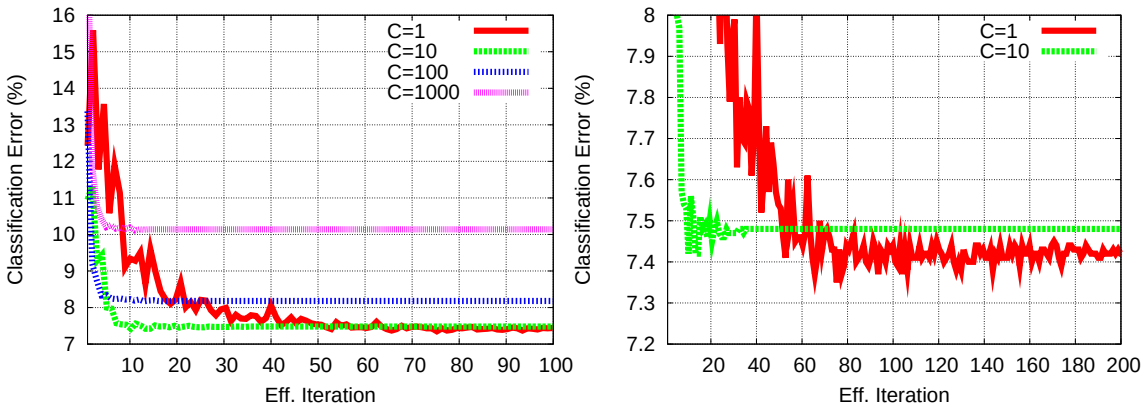


Figure 4: Validation error results on the MNIST learning task for log-linear models trained using the EG randomized online algorithm. The X axis shows the number of effective iterations over the entire data set. The Y axis shows validation error percentages. The left figure shows plots for values of C equal to 1, 10, 100, and 1000. The right figure shows plots for C equal to 1 and 10 at a larger scale.

$C = 10$ is a reasonable approximation to the solution for $C = 1$, and is considerably easier to solve; in the annealing strategy we start with an easier problem and then gradually move towards the harder problem of $C = 1$.

7.1.2 AN EFFICIENT METHOD FOR OPTIMIZING A RANGE OF C VALUES

In practice, when estimating parameters using either regularized log-likelihood or hinge-loss, a range of values for C are tested, with cross-validation or validation on a held-out set being used to choose the optimal value of C. In the previously described experiments, we independently optimized log-likelihood-based models for different values of C. In this section we describe a highly efficient method for training a sequence of models for a range of values of C.

The method is as follows. We pick some maximum value for C; as in our previous experiments, we will choose a maximum value of $C = 1000$. We also pick a tolerance value ε, and a parameter $0 < \nu < 1$. We then optimize C using the randomized online algorithm, until the duality gap is less than $\varepsilon \times p$, where p is the primal value. Once the duality gap has converged to within this ε tolerance, we reduce C by a factor of ν, and again optimize to within an ε tolerance. We continue this strategy—for each value of C optimizing to within a factor of ε, then reducing C by a factor of ν—until C has reached a low enough value. At the end of the sequence, this method recovers a series of models for different values of C, each optimized to within a tolerance of ε.

It is crucial that each time we decrease C, we take our initial dual values to be the final dual values resulting from optimization for the previous value of C. In practice, if C does not decrease too quickly, the previous dual values are a very good starting point for the new value of C; this corresponds to a "warm start" in optimizing values of C that are less than the maximum value. A

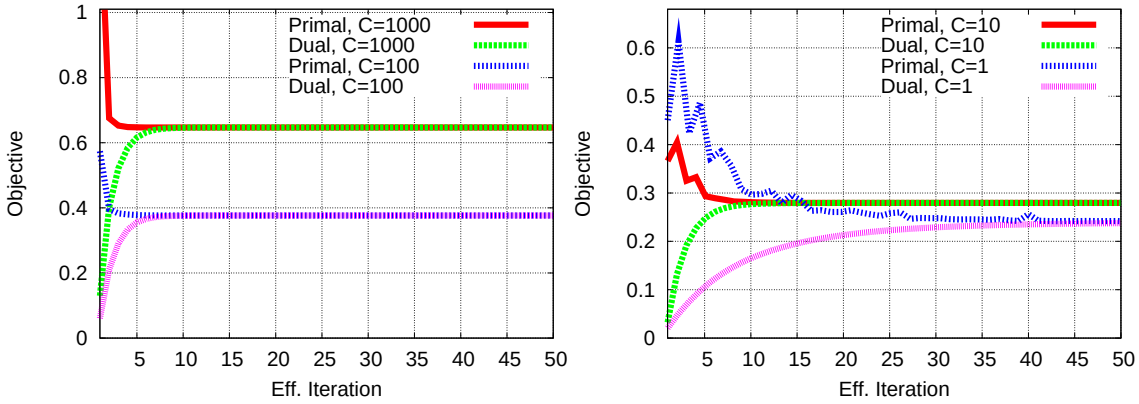


Figure 5: Primal and dual objective values on the MNIST learning task for log-linear models trained using the EG randomized online algorithm. The dual values have been negated so that the primal and dual problems have the same optimal value. The X axis shows the number of effective iterations over the entire data set. The Y axis shows the value of the primal or dual objective functions. The left figure shows plots for values of C equal to 1000 and 100; the right figure shows plots for C equal to 10, and 1. In all cases the primal and dual objectives converge to the same value, with faster convergence for larger values of C.

similar initialization method is used in Koh et al. (2007) in the context of ℓ_1 regularized logistic regression.

As one example of this approach, we trained models in this way with the starting (maximum) value of C set to 1000, ε set to 0.001 (i.e., 0.1%), and v set to 0.7. Table 2 shows the number of iterations of training required for each value of C. The benefits of using the previous dual values at each new value of C are clear: for $13.84 \leq C \leq 700$ at most 5 iterations are required for convergence; even for $C = 0.798$ only 15.24 iterations are required; a range of 25 different values of C between 1000 and 0.274 can be optimized with 211.17 effective iterations over the training set.

7.1.3 COMPARISONS TO STOCHASTIC GRADIENT DESCENT

This section compares performance of the EG algorithms to stochastic gradient descent (SGD) on the primal objective. In SGD the parameters $\mathbf{w}$ are initially set to be 0. At each step an example index i is chosen at random, and the following update is performed:

$$\mathbf{w} = \mathbf{w} - \eta \frac{\partial}{\partial \mathbf{w}} \left(-\log p(y_i \,|\, x_i; \mathbf{w}) + \frac{C}{2n} \|\mathbf{w}\|^2 \right) ,$$

where $\eta > 0$ is a learning rate. The term

$$\frac{\partial}{\partial \mathbf{w}} \left(-\log p(y_i \,|\, x_i; \mathbf{w}) + \frac{C}{2n} \|\mathbf{w}\|^2 \right) ,$$

C	Iterations	Total Iterations	Error
1000	11	11	0.1011
700	3	14	0.0968
490	4.01	18.01	0.0926
343	4.09	22.1	0.0895
240.1	4.24	26.34	0.0869
168.07	4.32	30.67	0.0846
117.649	4.3	34.97	0.0829
82.3543	4.29	39.27	0.0809
57.648	4.32	43.6	0.0803
40.3536	4.33	47.93	0.0775
28.2475	4.34	52.28	0.0768
19.7733	4.36	56.64	0.0758
13.8413	4.38	61.03	0.076
9.6889	5.47	66.51	0.0744
6.78223	5.49	72	0.0741
4.74756	5.51	77.52	0.0732
3.32329	6.6	84.12	0.0736
2.32631	7.69	91.82	0.0735
1.62841	8.78	100.61	0.0729
1.13989	12	112.62	0.074
0.797923	15.24	127.86	0.0747
0.558546	20.61	148.47	0.0749
0.390982	27.05	175.53	0.074
0.273687	35.63	211.17	0.0747

Table 2: Table showing number of effective iterations required to optimize a sequence of values for C for the MNIST task, using the method described in Section 7.1.2. The column C shows the sequence of decreasing regularizer constants. *Iterations* shows the number of effective iterations over the training set required to optimized each value of C. *Total iterations* shows the cumulative value of *Iterations*, and *Error* shows the validation error obtained for every C value. It can be seen that the optimal error is reached at $C = 1.62841$.

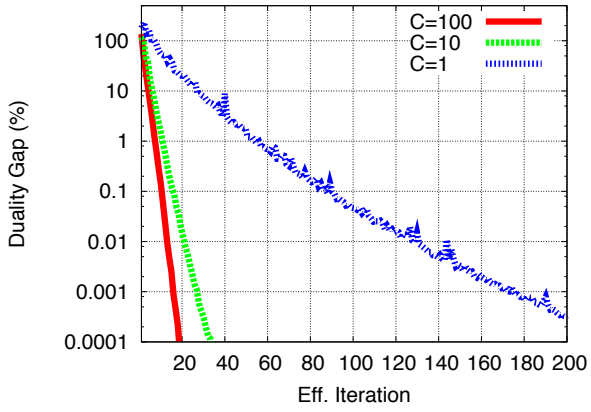


Figure 6: Graph showing the duality gap on the MNIST learning task for log-linear models trained using the EG randomized online algorithm. The X axis shows the number of effective iterations over the entire data set. The Y axis (with a log scale) shows the value of the duality gap, as a percentage of the final optimal value.

can be thought of as an estimate of the gradient of the primal objective function for the entire training set.

In our experiments, we chose the learning rate η to be

$$\eta = \frac{\eta_0}{1 + k/n} ,$$

where $\eta_0 > 0$ is a constant, n is the number of training examples, and k is the number of updates that have been performed up to this point. Thus the learning rate decays to 0 with the number of examples that are updated. This follows the approach described in LeCun et al. (1998); we have consistently found that it performs better than using a single, fixed learning rate.

We tested SGD for C values of 1000, 100, 10, 1, 0.1 and 0.01. In each case we chose the value of η_0 as follows. For each value of C we first tested values of η_0 equal to 1, 0.1, 0.01, 0.001, and 0.0001, and then chose the value of η_0 which led to the best validation error after a single iteration of SGD. This strategy resulted in a choice of $\eta_0 = 0.01$ for all values of C except $C = 1000$, where $\eta_0 = 0.001$ was chosen. We have found this strategy to be a robust method for choosing η_0 (note that we do not want to run SGD for more than one iteration with all (C, η_0) combinations, since each iteration is costly).

Figure 9 compares validation error rates for SGD and the randomized EG algorithm. For the initial (roughly 5) iterations of training, SGD has better validation error scores, but beyond this the EG algorithm is very competitive on this task. Note that the amount of computation for SGD does not include the iterations required to find the optimal value of η_0; if this computation was included the SGD curves would be shifted 5 iterations to the right.

Figure 10 shows graphs comparing the primal objective value for EG and SGD. For C equal to 1000, 100, and 10, the results are similar: SGD is initially better than EG, but after around 5

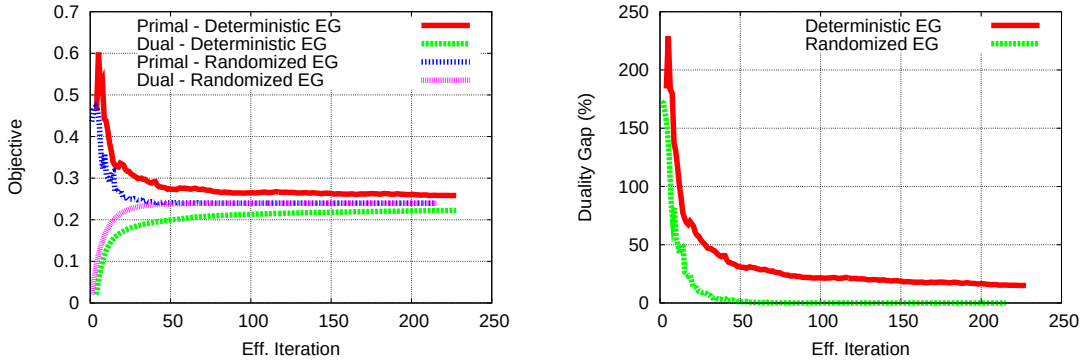


Figure 7: Results on the MNIST learning task, comparing the randomized and deterministic online EG algorithms, for $C = 1$. The left figure shows primal and dual objective values for both algorithms. The right figure shows the normalized value of the duality gap: $(\text{primal(t)} - \text{dual(t)})/\text{opt}$, where opt is the value of the joint optimum of the primal and dual problems, and t is the iteration number. The X axis counts the number of effective iterations over the entire data set.

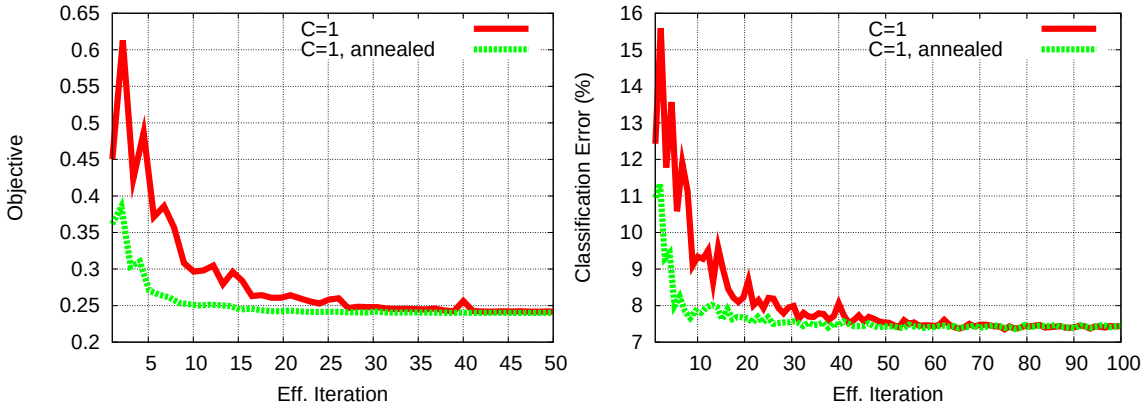


Figure 8: Results on the MNIST learning task, for $C = 1$, comparing the regular EG randomized algorithm with an annealed version of the algorithm (see Section 7.1.1). The left figure shows primal objective values calculated for $C = 1$; the right figure shows validation error. The annealed strategy gives significantly faster convergence.

iterations EG overtakes SGD, and converges much more quickly to the optimal point. The difference between EG and SGD appears to become more pronounced as C becomes smaller. For $C = 1$ our

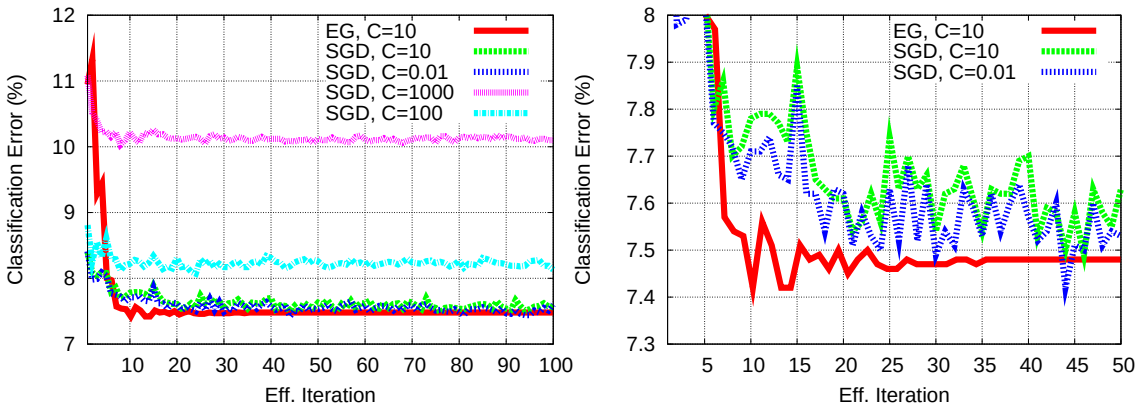


Figure 9: Graphs showing validation error results on the MNIST learning task, comparing the EG randomized algorithm to stochastic gradient descent (SGD). The X axis shows number of effective training iterations, the Y axis shows validation error in percent. The EG results are shown for $C = 10$; SGD results are shown for several values of C. For SGD for $C = 1$, $C = 0.1$, and $C = 0.01$ the curves were nearly identical, hence we omit the curves for $C = 1$ and $C = 0.1$. Note that the amount of computation for SGD does not include the iterations required to find the optimal value for the learning rate η_0.

strategy for choosing η_0 does not pick the optimal value for η_0 at least when evaluating the primal objective; see the caption to the figure for more discussion. EG again appears to out-perform SGD after the initial few iterations.

7.1.4 COMPARISONS TO L-BFGS

One of the standard approaches to training log-linear models is using the L-BFGS gradient-based algorithm (Sha and Pereira, 2003). L-BFGS is a batch algorithm, in the sense that its updates require evaluating the primal objective and gradient, which involves iterating over the entire data-set. To compare L-BFGS to EG, we used the implementation based on Byrd et al. (1995).[13]

For L-BFGS, a total of n training examples must be processed every time the gradient or objective function is evaluated; note that because L-BFGS uses a line search, each iteration may involve several such evaluations.[14]

13. Specifically, we used the code by Zhu, Byrd, Lu, and Nocedal (www.ece.northwestern.edu/∼nocedal/) with L. Stewart's wrapper (www.cs.toronto.edu/∼liam/). In all the experiments, we used 10 pairs of saved gradient vectors (see also Sha and Pereira, 2003).

14. The implementation of L-BFGS that we use requires both the gradient and objective when performing the line-search. In some line-search variants, it is possible to use only objective evaluations. In this case, the EG line search will be somewhat more costly, since the dual objective requires evaluations of both marginals and partition function, whereas the primal objective only requires the partition function. This will have an effect on running times only if the EG line search evaluates more than one point, which happened for less than 10%.

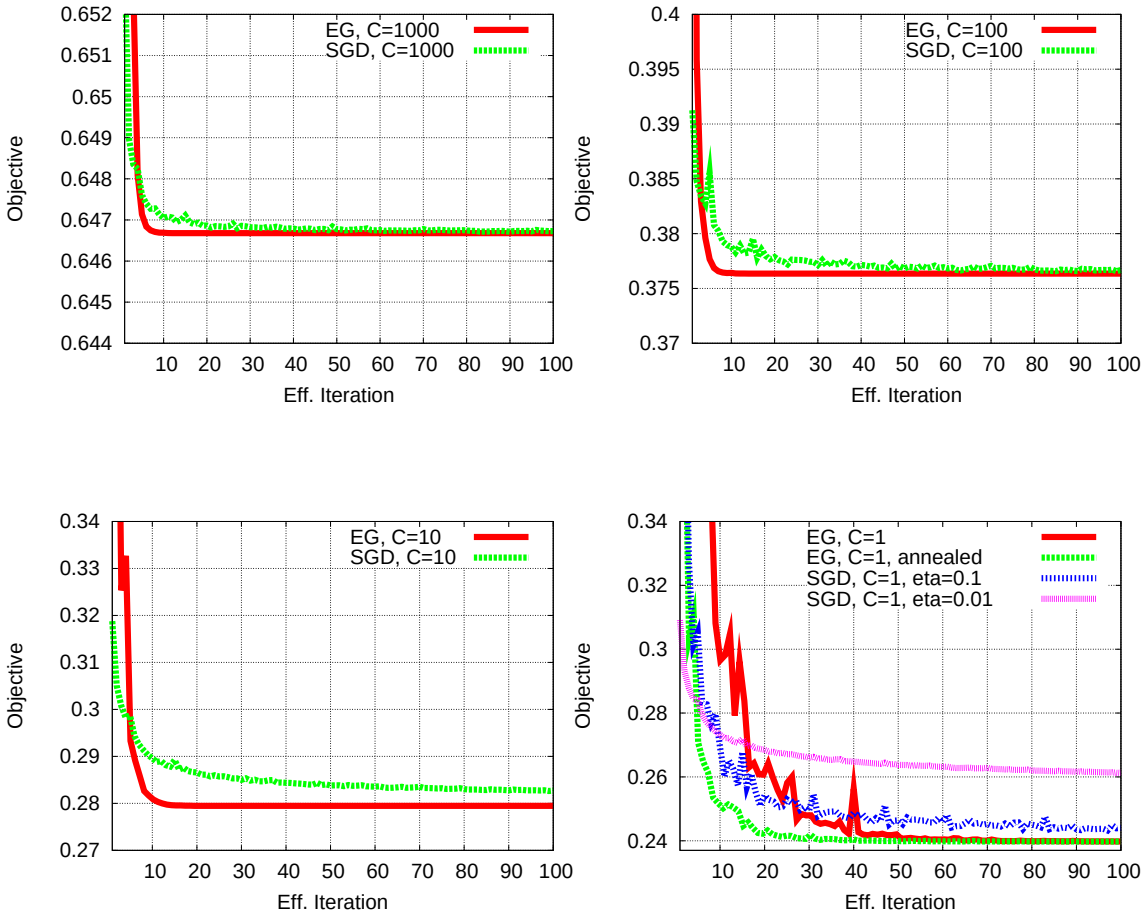


Figure 10: Graphs showing primal objective values on the MNIST learning task, comparing the EG randomized algorithm to stochastic gradient descent (SGD). The X axis shows number of effective training iterations, the Y axis shows primal objective. The graphs are for C equal to 1000, 100, 10, and 1. For $C = 1$ we show EG results with and without the annealed strategy described in Section 7.1.1. For $C = 1$ we also show two SGD curves, for learning rates 0.01 and 0.1: in this case $\eta_0 = 0.01$ was the best-performing learning rate after one iteration for both validation error and primal objective, however a post-hoc analysis shows that $\eta_0 = 0.1$ converges to a better value in the limit. Thus our strategy for choosing η_0 was not optimal in this case, although it is difficult to know how $\eta_0 = 0.1$ could be chosen without post-hoc analysis of the convergence for the different values of η_0. For other values of C our strategy for picking η_0 was more robust.

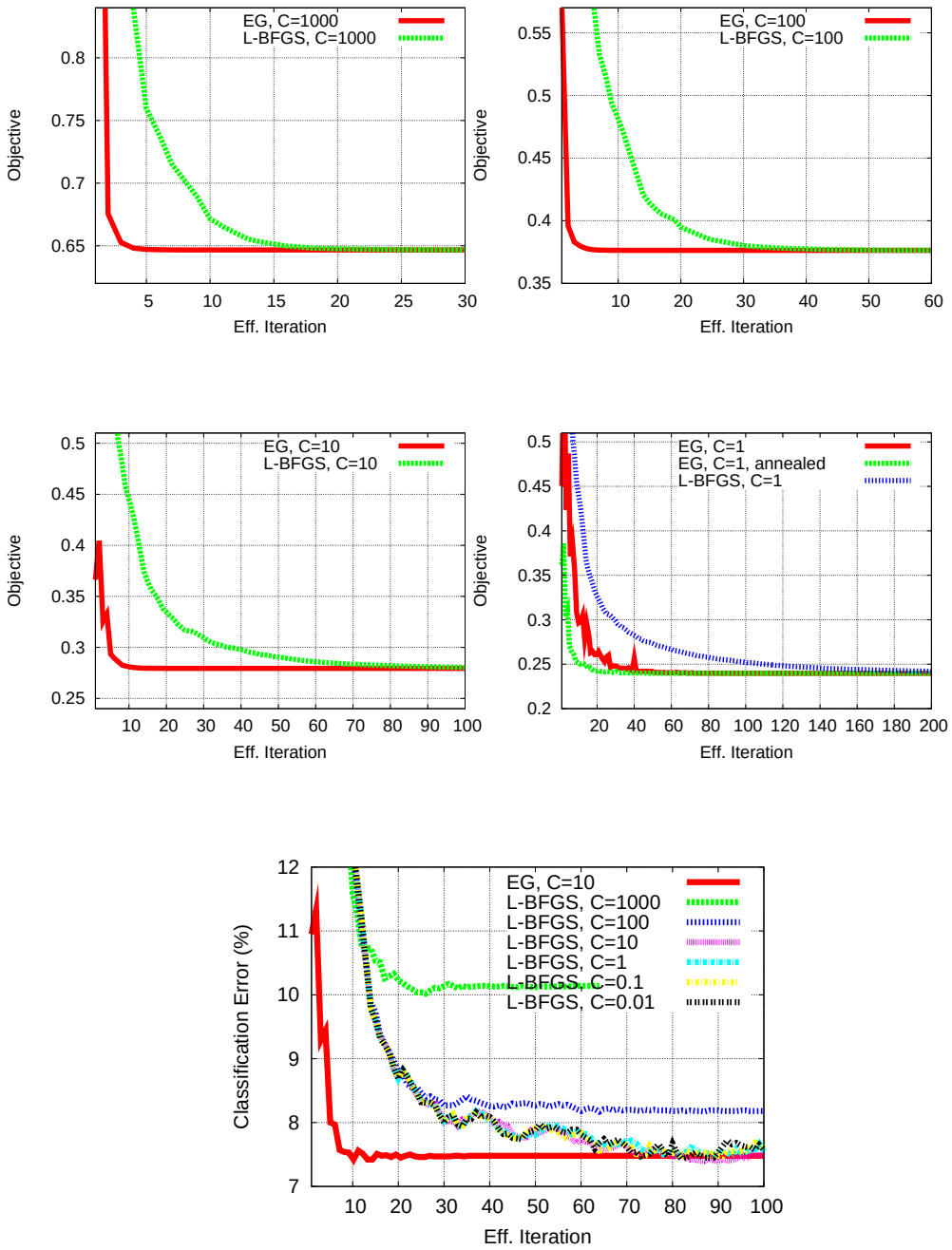


Figure 11: Results on the MNIST learning task, comparing the EG algorithm to L-BFGS. The figures on the first and second row show the primal objective for both algorithms, for various values of C. The bottom curve shows validation error for L-BFGS for various values of C and for EG with $C = 10$.

As in Section 7.1.3, we calculated primal values for EG. Figure 11 shows the primal objective for EG, and L-BFGS. It can be seen that the primal value for EG converges considerably faster than the L-BFGS one. Also shown is a curve of validation error for both algorithms. Here we show the results for EG with $C = 10$ and L-BFGS with various C values. It can be seen that L-BFGS does not outperform the EG curve for any value of C.

7.2 Structured learning - Dependency Parsing

Parsing of natural language sentences is a challenging structured learning task. Dependency parsing (McDonald et al., 2005) is a simplified form of parsing where the goal is to map sentences x into projective directed spanning trees over the set of words in x. Each label y is a set of directed arcs (dependencies) between pairs of words in the sentence. Each dependency is a pair (h,m) where h is the index of the head word of the dependency, and m is the index of the modifier word. Assuming we have a function $\mathbf{f}(x,h,m)$ that assigns a feature vector to dependencies (h,m), we can use a weight vector $\mathbf{w}$ to score a given tree y by $\mathbf{w}\cdot\sum_{(h,m)\in y}\mathbf{f}(x,h,m)$. Dependency parsing corresponds to a structured problem where the parts r are dependencies (h,m); the approach described in Section 4 can be applied efficiently to dependency structures. For projective dependency trees (e.g., see Koo et al., 2007), the required marginals can be computed efficiently using a variant of the inside-outside algorithm (Baker, 1979).

In the experiments below we use a feature set $\mathbf{f}(x,h,m)$ similar to that in McDonald et al. (2005) and Koo et al. (2007), resulting in $2,500,554$ features. We report results on the Spanish data-set which is part of the CoNLL-X Shared Task on multilingual dependency parsing (Buchholz and Marsi, 2006). The training data consists of $2,306$ sentences ($58,771$ tokens). To evaluate validation error, we use $1,000$ sentences ($30,563$ tokens) and report accuracy (rate of correct edges in a predicted parse tree) on these sentences.[15] Since we used only sentences from the training set, results are not directly comparable to the CoNLL-X shared task results. However, our previous work on this data set (Koo et al., 2007) shows that regularized max-margin and log-linear models typically outperform the averaged perceptron, which is not explicitly regularized.

As in the multi-class experiments, we compare to SGD and L-BFGS. The implementation of the algorithms is similar to that described in Section 7.1. The gradients for SGD and L-BFGS were obtained by calculating the relevant marginals of the model, using the inside-outside algorithm that was also used for EG. The learning rate for SGD was chosen as in the previous section; that is, we tested several learning rates ($\eta_0 = 1, 0.1, 0.001, 0.0001$) and chose the one that yielded the minimum validation error after one iteration.

Figure 12 shows results for EG and L-BFGS on the parsing task. We experiment with values of C in the set $\{0.1, 1, 10, 100, 1000\}$. Of these, the value that results in optimal validation error was $C = 10$. The performance of L-BFGS, SGD and EG is demonstrated in terms of the primal objective for a subset of the C values. L-BFGS and EG both converge to the optimal value, and EG is significantly faster. On the other hand, SGD does not converge to the optimum for all C values (e.g., for $C = 1, 10$), and when it does converge to the optimum, it is slower than EG.

Figure 12 also shows the validation error for EG at the optimal C value, compared to validation error for L-BFGS and SGD at various C values. Again, it can be seen that EG significantly outperforms L-BFGS. For SGD, performance is comparable to EG. However, as mentioned earlier, SGD

15. All $3,306$ sentences were obtained from the training data section of the CoNLL-X Spanish data-set (Civit and Martí, 2002).

C	Iterations	Total Iterations	Accuracy
1000	8	8	72.44
700	3.01	11.01	73.42
490	4.76	15.77	74.35
343	4.9	20.67	75.29
240.1	4.91	25.58	76.13
168.07	6.1	31.68	77.13
117.649	6.06	37.74	77.82
82.354	6.08	43.82	78.74
57.648	7.23	51.05	79.41
40.353	7.23	58.28	79.99
28.247	8.33	66.61	80.38
19.773	9.4	76.01	80.60
13.841	12.6	88.61	80.77
9.688	13.71	102.32	80.72
6.782	19.03	121.35	80.67
4.747	23.33	144.68	80.61
3.323	30.82	175.5	80.31
2.326	37.22	212.72	80.18
1.628	45.8	258.52	79.98
1.139	57.53	316.05	79.63
0.797	73.6	389.65	79.36

Table 3: Table showing number of effective iterations required to optimize a sequence of values for C for the parsing task, using the method described in Section 7.1.2. The column C shows the sequence of decreasing regularizer constants. *Iterations* shows the number of effective iterations over the training set required to optimize each value of C. *Total iterations* shows the cumulative value of *Iterations*, and *Accuracy* shows the validation accuracy obtained for every C value. It can be seen that the optimal accuracy is reached at $C = 13.841$.

in fact does not successfully optimize the primal objective for low values of C, and for higher values of C the SGD primal objective is slower to converge.

As in the multi-class experiments (see Figure 10), it is possible to find learning rates for SGD such that it converges to the primal optimum for $C = 1, 10$. However, the optimality of these rates only becomes evident after 10 iterations or more (results not shown). Thus, to find a learning rate for SGD that actually solves the optimization problem would typically require an additional few tens of iterations, making it significantly slower than EG.

Finally, it is possible to use EG to efficiently optimize over a set of regularization constants, as in Section 7.1.2. Table 3 shows results for a sequence of regularization constants.

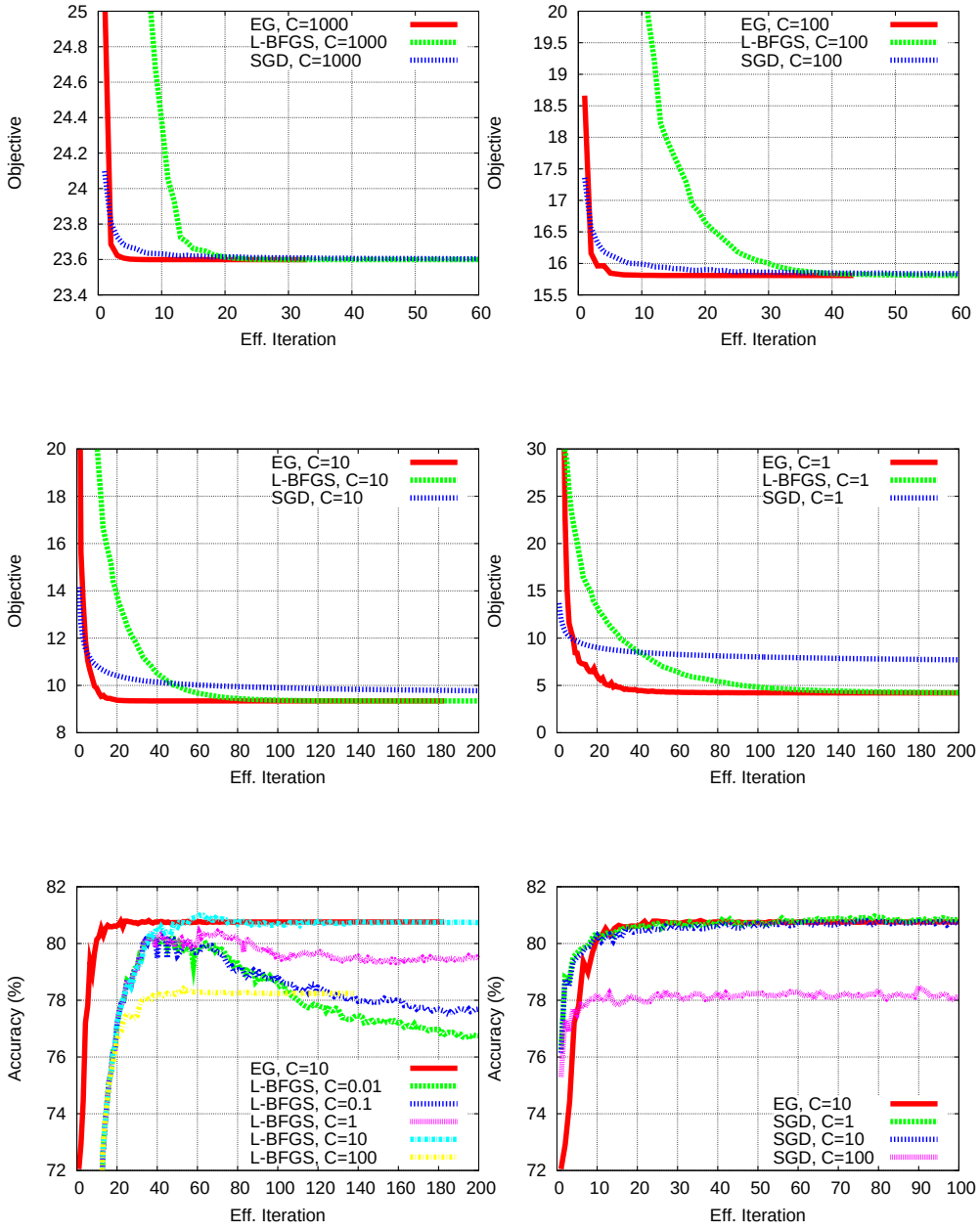


Figure 12: Results on the dependency-parsing task, comparing the EG algorithm to L-BFGS and SGD. All algorithms are trained on the log-linear objective function. The figures on the first and second rows show the primal objective for the three algorithms, for various values of C. The left bottom plot shows validation accuracy (measured as the fraction of correctly predicted edges) for L-BFGS for various values of C and for EG with $C = 10$. The right bottom plot show validation accuracy for EG (with $C = 10$) and SGD.

8. Experiments on Max-Margin Models

The max-margin loss (Eq. 3) has a discontinuity in its derivative. This makes optimization of max-margin models somewhat more involved than log-linear ones, since gradient algorithms such as L-BFGS cannot be used. This difficulty is exacerbated in the case of structured prediction models, since maximization in Eq. 3 is potentially over an exponentially large set.

In this section, we apply the EG algorithm to the max-margin problem, and compare its performance to the SVM-Struct algorithm presented in Tsochantaridis et al. (2004).[16] SVM-Struct is based on a cutting-plane algorithm that operates on the dual max-margin problem (D-MM) and results in monotone improvement in this dual. In this sense, it is similar to our EG algorithm. In order to facilitate a fair comparison, we report the performance of the two algorithms as a function of time. We do not report results by iteration since EG and SVM-struct involve different computation per iteration (e.g., SVM-Struct solves a QP per iteration).

We applied SVM-Struct and EG to the dependency parsing problem described in Section 7.2. To apply SVM-Struct to this problem, we supply it with a routine that finds the $y \in \mathcal{Y}$ which attains the maximum of the hinge-loss in Eq. 3. This maximum can be found using a Viterbi-style algorithm. For the value of C we experimented with $C \in \{1, 10, 100, 1000, 10000\}$. The optimal value in terms of validation error was $C = 100$.

Figure 13 shows results in terms of primal and dual objective and in terms of accuracy. It can be seen that EG is considerably faster than SVM-Struct for most C values. The performance is comparable only for $C = 1$, where convergence is slow for both algorithms.

9. Conclusion

We have presented novel algorithms for large-scale learning of log-linear and max-margin models, which provably converge to the optimal value of the respective loss functions. Although the algorithms have both batch and online variants, the online version turns out to be much more effective, both in theory and in practice. Our theoretical results (see Section 5.1) suggest that the online algorithm requires a factor of n less iterations to achieve a desired accuracy ε in the dual objective. This factor results from the fact that the online algorithm can use a learning rate η that is n times larger than the batch case to obtain updates that decrease the dual objective. Intuitively, this difference is associated with the fact that the batch algorithm updates all $\mathbf{u}$ values simultaneously. The dual objective has a term $\mathbf{u}^T A\mathbf{u}$ which involves all the $\mathbf{u}_i$ variables and second order interactions between them. It turns out that for batch updates only a relatively small change in the $\mathbf{u}_i$ is allowed, if one still requires an improvement in the dual objective after the update. It is possible that our bounds for the batch convergence rate are more conservative than those for the online case. However, we have observed in practice that the batch algorithm is much slower to converge. Furthermore, we also observed that other batch-based algorithms such as L-BFGS and conjugate gradient converge more slowly than the online EG algorithm.

Our results provide an $O(\log(\frac{1}{\varepsilon}))$ rate for the log-linear model, as opposed to $O(\frac{1}{\varepsilon})$ for max-margin. If these bounds are tight, they would imply that log-linear models have an advantage over max-margin ones in terms of training efficiency. However, it is possible that the analysis is not tight, and that improved rates may also be obtained for the max-margin model. In any case, this raises

16. The code is available from svmlight.joachims.org/svm_struct.html.

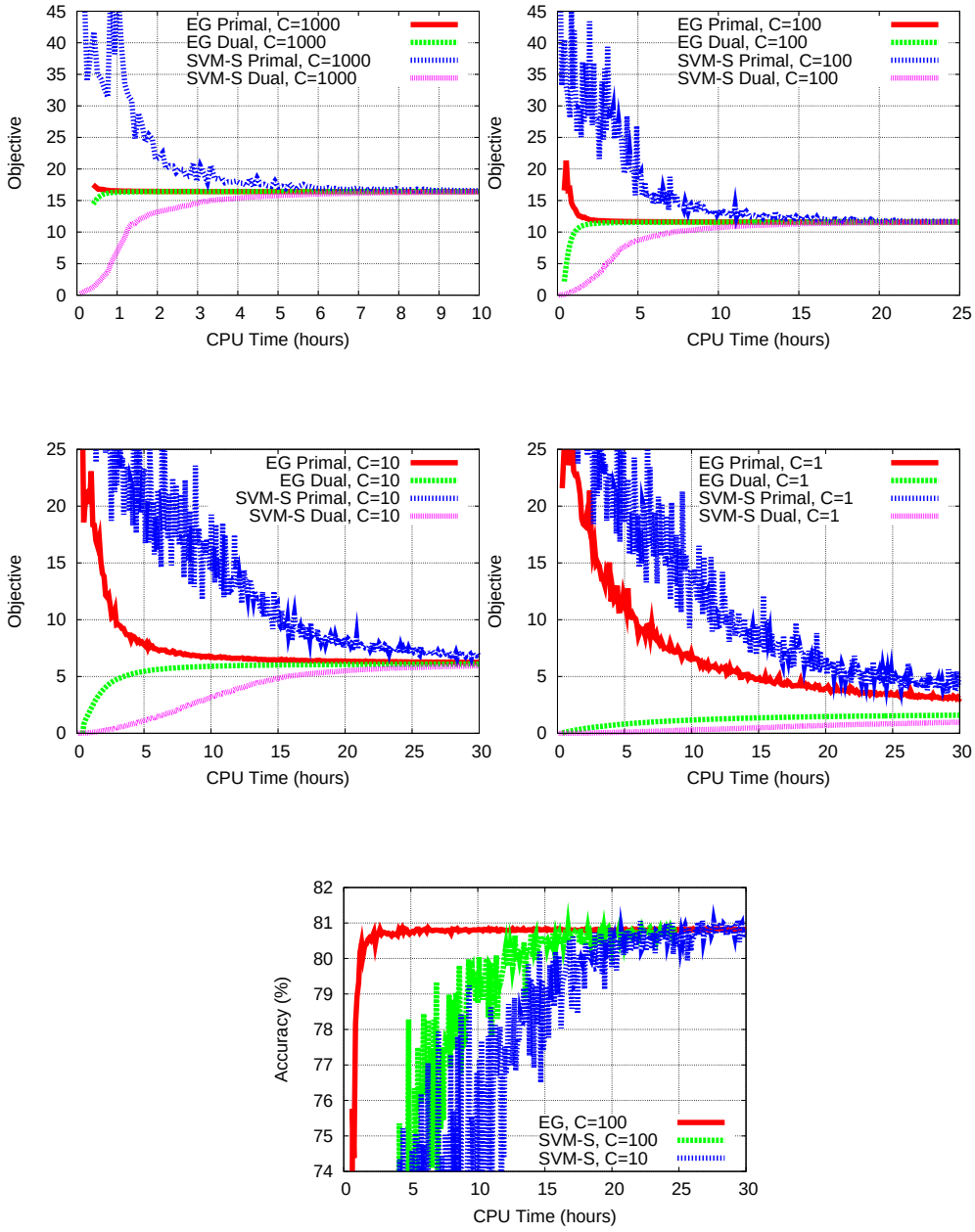


Figure 13: Results on the dependency-parsing task, comparing the EG algorithm to SVM-Struct. Both algorithms are trained on a max-margin model. The figures on the first and second rows show the primal objective for both algorithms, for various values of C. The bottom curve shows validation accuracy (measured as the fraction of correctly predicted edges) for SVM-Struct for various values of C and for EG with $C = 100$ (the value that yielded the highest validation accuracy). The X axis on all curves is running time in hours.

the interesting question of comparing classification models not only in terms of accuracy but also in terms of optimization efficiency.

Our convergence rates are with respect to accuracy in the dual objective. Some previous work (e.g., Shalev-Shwartz et al., 2007) has considered the accuracy with respect to the primal objective. It is relatively easy to show that in order to obtain ε accuracy in the primal, the EG algorithms require $O(\log(\frac{1}{\varepsilon}))$ updates for the log-linear problem and $O(\frac{1}{\varepsilon^2})$ for the max-margin case. It is possible that a more refined analysis of the max-margin case will result in $O(\frac{1}{\varepsilon})$ (e.g., see List et al., 2007), but we leave this for further study.

Most of our proofs rely on a relation between B_Q and the KL divergence. This relation holds for max-margin learning as well, a fact that simplifies previous results in this setting (Bartlett et al., 2005). We expect a similar analysis to hold for other functions Q.

An interesting extension of our method is to using second order derivative information, or its approximations, as in L-BFGS (Byrd et al., 1995). Such information may be used to obtain more accurate minimization for each $\mathbf{u}_i$ and may speed up convergence. Another possible improvement is to the line search method. In the experiments reported here we use a crude mechanism for adapting the learning rate, and it is possible that a more careful procedure will improve convergence rates in practice.

Parallelization is becoming increasingly relevant as multi-core CPUs become available. For the batch EG algorithm, it is straightforward to distribute the computation among k processors. One method for distributing the online EG algorithm would be to update k examples in parallel on k different processors. It should be possible to analyze this setting in a similar way to our proofs for the online case, but we leave this to future work.

Finally, our results show that the EG algorithms are highly competitive with state-of-the-art methods for training log-linear and max-margin models. We thus expect them to become useful as learning algorithms, particularly in the structured prediction setting.

Acknowledgments

The authors gratefully acknowledge the following sources of support. Amir Globerson was supported by a fellowship from the Rothschild Foundation - Yad Hanadiv. Terry Koo was funded by a grant from the NSF (DMS-0434222) and a grant from NTT, Agmt. Dtd. 6/21/1998. Xavier Carreras was supported by the Catalan Ministry of Innovation, Universities and Enterprise, and by a grant from NTT, Agmt. Dtd. 6/21/1998. Michael Collins was funded by NSF grants 0347631 and DMS-0434222. Peter Bartlett was funded by a grant from the NSF (DMS-0434383).

Appendix A. $O(\frac{1}{\varepsilon})$ Rate for Batch Algorithms - Proof of Lemma 5

We use a similar proof technique to that of Kivinen and Warmuth (2001). In particular, we need the following Lemma, which is very similar to results used by Kivinen and Warmuth (2001):

Lemma 10 *(See Kivinen and Warmuth (2001), Proof of Lemma 4) For any convex function $Q(\mathbf{u})$ over Δ^n, for any $\mathbf{z} \in \Delta^n$, and any $\mathbf{u}^t$ in the interior of Δ^n, if $\mathbf{u}^{t+1}$ is derived from $\mathbf{u}^t$ using the EG updates with a learning rate η, then*

$$\eta Q(\mathbf{u}^t) - \eta Q(\mathbf{z}) = D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] + D[\mathbf{u}^t\|\mathbf{u}^{t+1}] - \eta B_Q[\mathbf{z}\|\mathbf{u}^t] \; . \tag{9}$$

Proof: By the definition of Bregman divergence, we have

$$\eta Q(\mathbf{u}^t) - \eta Q(\mathbf{z}) = -\eta \nabla Q(\mathbf{u}^t) \cdot (\mathbf{z} - \mathbf{u}^t) - \eta B_Q[\mathbf{z} \| \mathbf{u}^t] \ . \tag{10}$$

Given that $\mathbf{u}^{t+1}$ is derived from $\mathbf{u}^t$ using EG updates,

$$u_{i,y}^{t+1} = \frac{u_{i,y}^t e^{-\eta \nabla_{i,y}}}{Z_i^t} \ ,$$

where Z_i^t is a normalization constant, and $\nabla_{i,y} = \frac{\partial Q(\mathbf{u}^t)}{\partial u_{i,y}}$. Simple algebra then shows that:

$$\begin{aligned} D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] + D[\mathbf{u}^t\|\mathbf{u}^{t+1}] &= \sum_{i,y} \left(z_{i,y} \log \frac{z_{i,y}}{u_{i,y}^t} - z_{i,y} \log \frac{z_{i,y}}{u_{i,y}^{t+1}} + u_{i,y}^t \log \frac{u_{i,y}^t}{u_{i,y}^{t+1}} \right) \\ &= \sum_{i,y} (z_{i,y} - u_{i,y}^t) \log \frac{u_{i,y}^{t+1}}{u_{i,y}^t} \\ &= \sum_{i,y} (z_{i,y} - u_{i,y}^t)(-\eta \nabla_{i,y} - \log Z_i^t) \\ &= \sum_{i,y} (z_{i,y} - u_{i,y}^t)(-\eta \nabla_{i,y}) \\ &= -\eta \nabla Q(\mathbf{u}^t) \cdot (\mathbf{z} - \mathbf{u}^t) \ . \end{aligned} \tag{11}$$

Note that we have used $\sum_{i,y}(z_{i,y} - u_{i,y}^t) \log Z_i^t = 0$, which follows because $\sum_{i,y} z_{i,y} \log Z_i^t = \sum_{i,y} u_{i,y}^t \log Z_i^t$ $= \sum_i \log Z_i^t$.

Combining Eq. 10 and Eq. 11 gives Eq. 9, thus proving the lemma. □

We can now prove Lemma 5:

Proof of Lemma 5: Using $-\eta B_Q[\mathbf{z}\|\mathbf{u}^t] \leq 0$, Lemma 10 implies that for all t

$$\eta Q(\mathbf{u}^t) - \eta Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] + D[\mathbf{u}^t\|\mathbf{u}^{t+1}] \ . \tag{12}$$

By the assumptions of Lemma 5, $Q(\mathbf{u})$ is τ-upper-bounded, and $0 \leq \eta \leq \frac{1}{\tau}$, hence by Lemma 3 we have

$$Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t\|\mathbf{u}^{t+1}] \ . \tag{13}$$

Combining Eqs. 12 and 13 gives for all t

$$\eta Q(\mathbf{u}^{t+1}) - \eta Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] \ .$$

Summing this over $t = 1, \ldots, T$ gives (the sum on the RHS telescopes)

$$\eta \sum_{t=1}^{T} Q(\mathbf{u}^{t+1}) - \eta T Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^1] - D[\mathbf{z}\|\mathbf{u}^{T+1}] \ .$$

Because $Q(\mathbf{u}^t)$ is monotone decreasing (by Eq. 13), we have $T Q(\mathbf{u}^{T+1}) \leq \sum_{t=1}^{T} Q(\mathbf{u}^{t+1})$ and simple algebra gives

$$Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{z}) + \frac{D[\mathbf{z}\|\mathbf{u}^1] - D[\mathbf{z}\|\mathbf{u}^{T+1}]}{\eta T} \ .$$

Dropping the term $D[\mathbf{z}\|\mathbf{u}^{T+1}]$ (because $-D[\mathbf{z}\|\mathbf{u}^{T+1}] \leq 0$) we obtain

$$Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{z}) + \frac{D[\mathbf{z}\|\mathbf{u}^1]}{\eta T} ,$$

as required.

Appendix B. $O(\log(\frac{1}{\varepsilon}))$ Rate for Batch Algorithms - Proof of Lemma 6

By the assumptions of Lemma 6, $Q(\mathbf{u})$ is τ-upper-bounded, and $0 \leq \eta \leq \frac{1}{\tau}$, hence by Lemma 3 we have for all t

$$Q(\mathbf{u}^t) - Q(\mathbf{u}^{t+1}) \geq \frac{1}{\eta} D[\mathbf{u}^t\|\mathbf{u}^{t+1}] .$$

Combining this result with Lemma 10 gives

$$\eta Q(\mathbf{u}^{t+1}) - \eta Q(\mathbf{z}) \leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] - \eta B_Q[\mathbf{z}\|\mathbf{u}^t] .$$

We can now make use of the assumption that $Q(\mathbf{u})$ is (μ,τ)-bounded, and hence $\eta B_Q[\mathbf{z}\|\mathbf{u}^t] \geq \eta\mu D[\mathbf{z}\|\mathbf{u}^t]$, to obtain

$$\begin{aligned} \eta Q(\mathbf{u}^{t+1}) - \eta Q(\mathbf{z}) &\leq D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] - \eta\mu D[\mathbf{z}\|\mathbf{u}^t] \\ &= (1-\eta\mu) D[\mathbf{z}\|\mathbf{u}^t] - D[\mathbf{z}\|\mathbf{u}^{t+1}] \\ &\leq (1-\eta\mu) D[\mathbf{z}\|\mathbf{u}^t] . \end{aligned} \tag{14}$$

If there exists a $t \leq T$ such that $Q(\mathbf{u}^{t+1}) - Q(\mathbf{z}) \leq 0$ then because $Q(\mathbf{u}^t)$ decreases monotonically with t we have $Q(\mathbf{u}^{T+1}) \leq Q(\mathbf{u}^{t+1}) \leq Q(\mathbf{z})$ and the lemma trivially holds. Otherwise, it must be the case that $Q(\mathbf{u}^{t+1}) - Q(\mathbf{z}) \geq 0$ for all $t \leq T$, and thus for all $t \leq T$

$$D[\mathbf{z}\|\mathbf{u}^{t+1}] \leq (1-\eta\mu) D[\mathbf{z}\|\mathbf{u}^t] .$$

Using this inequality recursively for $t = 1,\ldots,T$ we get

$$D[\mathbf{z}\|\mathbf{u}^{T+1}] \leq (1-\eta\mu)^T D[\mathbf{z}\|\mathbf{u}^1] .$$

Substituting back into Eq. 14 we obtain

$$Q(\mathbf{u}^{T+1}) - Q(\mathbf{z}) \leq \frac{(1-\eta\mu)^T}{\eta} D[\mathbf{z}\|\mathbf{u}^1] \leq \frac{e^{-\eta\mu T}}{\eta} D[\mathbf{z}\|\mathbf{u}^1] ,$$

where we have used $\log(1-x) \leq -x$.

Appendix C. Proof of Lemma 7

For the regularized log-likelihood dual, for any $\mathbf{v} \in \Delta$

$$Q_{\mathbf{u},i}(\mathbf{v}) = \sum_y v_y \log v_y + \frac{1}{2}\mathbf{v}^T A(i,i)\mathbf{v} + \sum_{j\neq i}\sum_y u_{j,y}\log u_{j,y} + \frac{1}{2}\sum_{j\neq i}\sum_{k\neq i}\mathbf{u}_j^T A(j,k)\mathbf{u}_k + \sum_{j\neq i}\mathbf{u}_j^T A(j,i)\mathbf{v} ,$$

where $A(j,k)$ is the $|\mathcal{Y}| \times |\mathcal{Y}|$ sub-matrix of A defined as $A_{y,z}(j,k) = A_{(j,y),(k,z)}$. To obtain the Bregman divergence $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}]$, note that the last three terms in $Q_{\mathbf{u},i}(\mathbf{v})$ are either constant or linear in $\mathbf{v}$ and thus do not contribute to $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}]$. It follows that

$$B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] = D[\mathbf{p}\|\mathbf{q}] + M_{A(i,i)}[\mathbf{p}\|\mathbf{q}] \ .$$

By a similar argument to the proof of Lemma 3, it follows that $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] \leq (1+|A(i,i)|_\infty)D[\mathbf{p}\|\mathbf{q}]$. Because $A(i,i)$ is a sub-matrix of A we have $|A(i,i)|_\infty \leq |A|_\infty$, and the first part of the lemma follows. For the max-margin dual, a similar argument shows that

$$B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] = M_{A(i,i)}[\mathbf{p}\|\mathbf{q}] \ ,$$

so we have $B_{Q_{\mathbf{u},i}}[\mathbf{p}\|\mathbf{q}] \leq |A(i,i)|_\infty D[\mathbf{p}\|\mathbf{q}] \leq |A|_\infty D[\mathbf{p}\|\mathbf{q}]$. □

Appendix D. Proof of Lemma 8

For the proof we will need some additional notation, which makes explicit the relationship between the sequence of dual variables $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{T+1}$ and the sequence of indices $k_1, k_2, \ldots, k_T$ used in the algorithm in Figure 2. We will use the following definitions:

- We use k_1^t to denote a sequence of indices $k_1, k_2, \ldots, k_t$. We take k_1^0 to be the empty sequence.
- We write $\mathbf{r} : \Delta^n \times [1 \ldots n] \to \Delta^n$ to denote the function that corresponds to an EG update on a single example. More specifically, we have

$$\begin{aligned} \mathbf{r}_i(\mathbf{u},k) &= \mathbf{u}_i \ \text{ for } i \neq k \\ r_{i,y}(\mathbf{u},k) &\propto u_{i,y} \exp\{-\eta \nabla_{i,y}\} \ \text{ where } \nabla_{i,y} = \tfrac{\partial Q(\mathbf{u})}{\partial u_{i,y}} \text{ for } i = k, \text{ for all } y \ . \end{aligned}$$

- Finally, for any choice of index sequence k_1^T we will define a sequence of dual variables using the following iterative definition:

$$\begin{aligned} \mathbf{u}(k_1^0) &= \mathbf{u}^1 \\ \mathbf{u}(k_1^t) &= \mathbf{r}(\mathbf{u}(k_1^{t-1}), k_t) \ \text{ for } t \geq 1 \ . \end{aligned}$$

  Here $\mathbf{u}^1$ is the initial setting of the dual variables, as shown in the algorithm in Figure 2.

From these definitions it follows that if k_1^T is the sequence of indices chosen during a run of the algorithm in Figure 2, then the sequence of dual variables $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{T+1}$ is such that $\mathbf{u}^{t+1} = \mathbf{u}(k_1^t)$ for $t = 0 \ldots T$. We can now give the proof.

Proof of Lemma 8. First, we have for any $\mathbf{u}, \mathbf{z} \in \Delta^n$

$$\begin{aligned} Q(\mathbf{u}) &\leq Q(\mathbf{z}) + (\mathbf{u}-\mathbf{z}) \cdot \nabla Q(\mathbf{u}) \\ &= Q(\mathbf{z}) + \frac{1}{\eta} \sum_{i=1}^n \Big[D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u},i)] + D[\mathbf{u}_i \| \mathbf{r}_i(\mathbf{u},i)] \Big] \ . \end{aligned} \tag{15}$$

The second line follows by similar arguments to those in the proof of Lemma 10.

Next consider the terms on the right-hand-side of the inequality. For any i, we have

$$\begin{aligned} & D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u}, i)] \\ &= D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u}, i)] + \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{u}_j] - \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{u}_j] \quad (16) \\ &= D[\mathbf{z}_i \| \mathbf{u}_i] - D[\mathbf{z}_i \| \mathbf{r}_i(\mathbf{u}, i)] + \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{u}_j] - \sum_{j=1\ldots n, j\neq i} D[\mathbf{z}_j \| \mathbf{r}_j(\mathbf{u}, i)] \quad (17) \\ &= D[\mathbf{z} \| \mathbf{u}] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}, i)] \ . \quad (18) \end{aligned}$$

Here Eq. 17 follows from Eq. 16 because $\mathbf{r}_j(\mathbf{u}, i) = \mathbf{u}_j$ for $j \neq i$. In addition, for any i we have

$$\frac{1}{\eta} D[\mathbf{u}_i \| \mathbf{r}_i(\mathbf{u}, i)] \leq Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u}, i)) \ . \quad (19)$$

This follows because by the assumption in the lemma, $Q(\mathbf{u})$ is τ-online-upper-bounded, so we have $B_{Q_{\mathbf{u},i}}[\mathbf{r}_i(\mathbf{u}, i) \| \mathbf{u}_i] \leq \tau D[\mathbf{r}_i(\mathbf{u}, i) \| \mathbf{u}_i]$. By an application of Lemma 2 to the convex function $Q_{\mathbf{u},i}$, noting that by assumption $\eta \leq 1/\tau$, it follows that

$$\frac{1}{\eta} D[\mathbf{u}_i \| \mathbf{r}_i(\mathbf{u}, i)] \leq Q_{\mathbf{u},i}(\mathbf{u}_i) - Q_{\mathbf{u},i}(\mathbf{r}_i(\mathbf{u}, i)) \ .$$

Finally, note that $Q_{\mathbf{u},i}(\mathbf{u}_i) = Q(\mathbf{u})$, and $Q_{\mathbf{u},i}(\mathbf{r}_i(\mathbf{u}, i)) = Q(\mathbf{r}(\mathbf{u}, i))$, giving the result in Eq. 19.

Combining Equations 15, 18 and 19 gives for any $\mathbf{u}$,

$$Q(\mathbf{u}) \leq Q(\mathbf{z}) + \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z} \| \mathbf{u}] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}, i)] \Big] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u}, i)) \Big] \ . \quad (20)$$

Because Eq. 20 holds for any value of $\mathbf{u}$, we have for all $t = 1 \ldots T$, for all $k_1^{t-1} \in [1 \ldots n]^{t-1}$,

$$\begin{aligned} Q(\mathbf{u}(k_1^{t-1})) \leq Q(\mathbf{z}) &+ \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z} \| \mathbf{u}(k_1^{t-1})] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}(k_1^{t-1}), i)] \Big] \\ &+ \sum_{i=1}^{n} \Big[Q(\mathbf{u}(k_1^{t-1})) - Q(\mathbf{r}(\mathbf{u}(k_1^{t-1}), i)) \Big] \ . \quad (21) \end{aligned}$$

We can now take an expectation of both sides of the inequality in Eq. 21. For any function $f(k_1^t)$, we use the notation $\mathbf{E}_t[f(k_1^t)]$ to denote the expected value of $f(k_1^t)$ when k_1^t is drawn uniformly at random from $[1, 2, \ldots, n]^t$; more precisely

$$\mathbf{E}_t[f(k_1^t)] = \frac{1}{n^t} \sum_{k_1^t \in [1,2,\ldots,n]^t} f(k_1^t) \ .$$

We apply the operator $\mathbf{E}_{t-1}$ to both sides of Eq. 21. We consider the different terms in turn. First,

$$\mathbf{E}_{t-1}[Q(\mathbf{u}(k_1^{t-1}))] = \mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] \ .$$

This follows because $Q(\mathbf{u}(k_1^{t-1}))$ does not depend on the values for $k_t, \ldots, k_T$. Clearly, $\mathbf{E}_{t-1}[Q(\mathbf{z})] = Q(\mathbf{z})$. Next,

$$\mathbf{E}_{t-1} \left[\frac{1}{\eta} \sum_{i=1}^{n} D[\mathbf{z} \| \mathbf{u}(k_1^{t-1})] - D[\mathbf{z} \| \mathbf{r}(\mathbf{u}(k_1^{t-1}), i)] \right]$$

$$
\begin{aligned}
&= \mathbf{E}_{t-1}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_{t-1}\left[\frac{n}{\eta}\sum_{i=1}^{n}\frac{1}{n}D[\mathbf{z}\|\mathbf{r}(\mathbf{u}(k_1^{t-1}),i)]\right] \\
&= \mathbf{E}_{t-1}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_{t}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t})]\right] \\
&= \mathbf{E}_{T}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_{T}\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t})]\right] .
\end{aligned}
$$

Finally, we consider the last term:

$$
\begin{aligned}
&\mathbf{E}_{t-1}\left[\sum_{i=1}^{n} Q(\mathbf{u}(k_1^{t-1})) - Q(\mathbf{r}(\mathbf{u}(k_1^{t-1}),i))\right] \\
&\quad= \mathbf{E}_{t-1}\left[nQ(\mathbf{u}(k_1^{t-1})) - n\sum_{i=1}^{n}\frac{1}{n}Q(\mathbf{r}(\mathbf{u}(k_1^{t-1}),i))\right] \\
&\quad= \mathbf{E}_{t-1}\left[nQ(\mathbf{u}(k_1^{t-1}))\right] - \mathbf{E}_{t}\left[nQ(\mathbf{u}(k_1^{t}))\right] \\
&\quad= \mathbf{E}_{T}\left[nQ(\mathbf{u}(k_1^{t-1}))\right] - \mathbf{E}_{T}\left[nQ(\mathbf{u}(k_1^{t}))\right] .
\end{aligned}
$$

Combining these results with Eq. 21 gives

$$
\begin{aligned}
\mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] \;\leq\; & Q(\mathbf{z}) + \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})]\right] - \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{t})]\right] \\
& + \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{t-1}))\right] - \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{t}))\right] . \qquad (22)
\end{aligned}
$$

Summing Eq. 22 over $t = 1 \ldots T$ gives

$$
\begin{aligned}
\sum_{t=1}^{T}\mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] \;\leq\; & TQ(\mathbf{z}) + \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{0})]\right] - \mathbf{E}_T\left[\frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{T})]\right] \\
& + \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{0}))\right] - \mathbf{E}_T\left[nQ(\mathbf{u}(k_1^{T}))\right] \\
\leq\; & TQ(\mathbf{z}) + \frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}(k_1^{0})] + n\left[Q(\mathbf{u}(k_1^{0})) - Q(\mathbf{u}^*)\right] \\
=\; & TQ(\mathbf{z}) + \frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}^1] + n\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] , \qquad (23)
\end{aligned}
$$

where $\mathbf{u}^* = \operatorname{argmin}_{\mathbf{u}\in\Delta^n} Q(\mathbf{u})$. Finally, note that for any value of k_1^T we have $Q(\mathbf{u}(k_1^t)) \leq Q(\mathbf{u}(k_1^{t-1}))$ for $t = 1 \ldots T$. Thus

$$
\mathbf{E}_T[Q(\mathbf{u}(k_1^t))] \leq \mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] ,
$$

and

$$
T\mathbf{E}_T[Q(\mathbf{u}(k_1^T)] \leq \sum_{t=1}^{T}\mathbf{E}_T[Q(\mathbf{u}(k_1^{t-1}))] .
$$

Combining this with Eq. 23 gives

$$
T\mathbf{E}_T[Q(\mathbf{u}(k_1^T)] \leq TQ(\mathbf{z}) + \frac{n}{\eta}D[\mathbf{z}\|\mathbf{u}^1] + n\left[Q(\mathbf{u}^1) - Q(\mathbf{u}^*)\right] ,
$$

thus proving the lemma. □

Appendix E. Proof of Lemma 9

(Note: this proof builds on notation and techniques given in the proof of Lemma 8, see Appendix D.)

We begin with the following identity

$$B_Q[\mathbf{z}||\mathbf{u}] \;=\; Q(\mathbf{z}) - Q(\mathbf{u}) - \nabla Q(\mathbf{u}) \cdot (\mathbf{z} - \mathbf{u}) \;.$$

Rearranging yields for any $\mathbf{z}, \mathbf{u} \in \Delta^n$,

$$\begin{aligned} Q(\mathbf{u}) - Q(\mathbf{z}) &= \nabla Q(\mathbf{u}) \cdot (\mathbf{u} - \mathbf{z}) - B_Q[\mathbf{z}||\mathbf{u}] \\ &= \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z}_i||\mathbf{u}_i] - D[\mathbf{z}_i||\mathbf{r}_i(\mathbf{u},i)] + D[\mathbf{u}_i||\mathbf{r}_i(\mathbf{u},i)] \Big] - B_Q[\mathbf{z}||\mathbf{u}] \;, \end{aligned}$$

where the second line follows by a similar argument to the proof of Lemma 10.

By applying similar arguments to those leading to Eq. 20 in Appendix D, we get for any $\mathbf{z}, \mathbf{u} \in \Delta^n$,

$$\begin{aligned} Q(\mathbf{u}) - Q(\mathbf{z}) &\leq \frac{1}{\eta} \sum_{i=1}^{n} \Big[D[\mathbf{z}\|\mathbf{u}] - D[\mathbf{z}\|\mathbf{r}(\mathbf{u},i)] \Big] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u},i)) \Big] - B_Q[\mathbf{z}||\mathbf{u}] \\ &= \frac{n}{\eta} D[\mathbf{z}\|\mathbf{u}] - \frac{n}{\eta} \sum_{i=1}^{n} \frac{1}{n} D[\mathbf{z}\|\mathbf{r}(\mathbf{u},i)] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u},i)) \Big] - B_Q[\mathbf{z}||\mathbf{u}] \\ &\leq \left(\frac{n}{\eta} - \mu \right) D[\mathbf{z}\|\mathbf{u}] - \frac{n}{\eta} \sum_{i=1}^{n} \frac{1}{n} D[\mathbf{z}\|\mathbf{r}(\mathbf{u},i)] + \sum_{i=1}^{n} \Big[Q(\mathbf{u}) - Q(\mathbf{r}(\mathbf{u},i)) \Big] \;, \end{aligned}$$

where in the third line, we have used the assumption from the lemma that $Q(\mathbf{u})$ is (μ, τ)-online-bounded, and hence $B_Q[\mathbf{z}||\mathbf{u}] \geq \mu D[\mathbf{z}\|\mathbf{u}]$. The inequality above holds for all $\mathbf{u}, \mathbf{z} \in \Delta^n$, so we can take $\mathbf{u} = \mathbf{u}(k_1^{t-1})$ for any sequence k_1^{t-1}. Taking expectations of both sides, and using similar arguments to those leading to Eq. 22 in Appendix D, we get

$$\begin{aligned} \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^{t-1})) \Big] - Q(\mathbf{z}) \;\leq\; & \left(\frac{n}{\eta} - \mu \right) \mathbf{E}_T \Big[D[\mathbf{z}\|\mathbf{u}(k_1^{t-1})] \Big] - \frac{n}{\eta} \mathbf{E}_T \Big[D[\mathbf{z}\|\mathbf{u}(k_1^{t})] \Big] \\ & + n \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^{t-1})) \Big] - n \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^{t})) \Big] \;, \end{aligned} \tag{24}$$

where $\mathbf{E}_T$ is again an expectation with respect to the sequence k_1^T being drawn from the uniform distribution over $[1 \ldots n]^T$. For convenience, define

$$\tilde{\mathcal{Q}}^t \equiv \mathbf{E}_T \Big[Q(\mathbf{u}(k_1^t)) \Big] - Q(\mathbf{z}) \qquad \text{and} \qquad \tilde{\mathcal{D}}^t \equiv \frac{1}{\eta} \mathbf{E}_T \Big[D[\mathbf{z}||\mathbf{u}(k_1^t)] \Big] \;.$$

We may assume that $\tilde{\mathcal{Q}}^t \geq 0$ for all $t \leq T$ since if this is not true the lemma trivially holds.[17] Eq. 24 can be rearranged to give

$$\begin{aligned} \tilde{\mathcal{Q}}^{t-1} &\leq (n - \eta\mu) \tilde{\mathcal{D}}^{t-1} - n \tilde{\mathcal{D}}^t + n \tilde{\mathcal{Q}}^{t-1} - n \tilde{\mathcal{Q}}^t \\ n \tilde{\mathcal{Q}}^t + n \tilde{\mathcal{D}}^t &\leq (n-1) \tilde{\mathcal{Q}}^{t-1} + (n - \eta\mu) \tilde{\mathcal{D}}^{t-1} \;\leq\; (n - \eta\mu) \left(\tilde{\mathcal{Q}}^{t-1} + \tilde{\mathcal{D}}^{t-1} \right) \\ \tilde{\mathcal{Q}}^t + \tilde{\mathcal{D}}^t &\leq \left(1 - \frac{\eta\mu}{n} \right) \left(\tilde{\mathcal{Q}}^{t-1} + \tilde{\mathcal{D}}^{t-1} \right) \;, \end{aligned} \tag{25}$$

17. Note that $\mathbf{E}_T \left[Q(\mathbf{u}(k_1^t)) \right]$ is monotone decreasing since every random sequence of updates results in monotone improvement. The lemma then holds by an argument similar to Appendix B.

where Eq. 25 uses the observation that $n-\eta\mu \geq n-1$ because $\eta\mu \leq 1$ (this follows because $\eta \leq 1/\tau$ and $\mu < \tau$ for some $\tau > 0$). By iterating this result, it follows that

$$\begin{aligned}
\tilde{\mathcal{Q}}^T + \tilde{\mathcal{D}}^T &\leq \left(1-\frac{\eta\mu}{n}\right)^T \left(\tilde{\mathcal{Q}}^0 + \tilde{\mathcal{D}}^0\right) \\
\mathbf{E}_T\left[Q(\mathbf{u}(k_1^T))\right] &\leq Q(\mathbf{z}) + \left(1-\frac{\eta\mu}{n}\right)^T \left(Q(\mathbf{u}^1) - Q(\mathbf{z}) + \frac{1}{\eta} D[\mathbf{z}||\mathbf{u}^1]\right) \\
&\leq Q(\mathbf{z}) + e^{-\eta\mu T/n} \left(Q(\mathbf{u}^1) - Q(\mathbf{z}) + \frac{1}{\eta} D[\mathbf{z}||\mathbf{u}^1]\right) ,
\end{aligned}$$

thus proving the lemma. □

Appendix F. Empirical Comparisons in Terms of Running Time

In this section we compare EG to SGD and L-BFGS in terms of running time. The experiments in the main text provide comparison in terms of "effective" iterations, which do not take into account the computational cost of processing a single example. Here we show that EG maintains its advantages over the other learning algorithms when running time is used as a performance measure, with similar relative improvements to those reported in the main text.

Clearly, any timed comparison depends on the quality of the implementations being compared. Data processing and gradient and objective calculations were performed using the same C++ code for all three algorithms: EG, SGD, and L-BFGS. For L-BFGS, we used the implementation based on Byrd et al. (1995).[18] This code is available online and is written in Fortran. The SGD update is straightforward and we implemented it ourselves in our C++ package. All the timing experiments were performed on a 1.8GHz AMD Opteron™ CPU.

We focus on the log-linear case here, since timing results for the max-margin case were provided in Section 8.

Figures 14 and 15 show results for the MNIST multi-class (see Section 7.1), and the parsing tasks (see Section 7.2) respectively. As in the results in the main text, it can be seen that the EG objective converges faster than the two other algorithms. Also, as in the main text, SGD converges quickly in terms of accuracy, but its objective converges very slowly to the optimum.

Note that the timing of the EG experiments includes the time required to convert the dual parameters to the primal representation. We have found that the EG algorithm is quite fast in practice; in the MNIST task, for example, the EG algorithm requires on average only 10% more time per iteration (including the step-size search) than SGD and L-BFGS. To help explain why EG is able to run almost as fast as SGD, Figure 16 presents pseudocode for the SGD and online EG algorithms. Both SGD and EG share the following operations: (a) inner products between the feature vectors and the primal vector, (b) computation of part-wise marginals, and (c) addition of scaled feature vectors to the primal vector. In the EG algorithm, we require two additional loops over $R(x_i)$ in order to update the dual variables and compute the dual entropy term. In practice, however, the cost of the two additional loops is dominated by the three shared operations mentioned above. Thus, processing a single example takes roughly the same time for EG and SGD. Similar arguments can be used to explain why EG can run almost as fast as L-BFGS.

18. Specifically, we used the code by Zhu, Byrd, Lu, and Nocedal (www.ece.northwestern.edu/∼nocedal/).

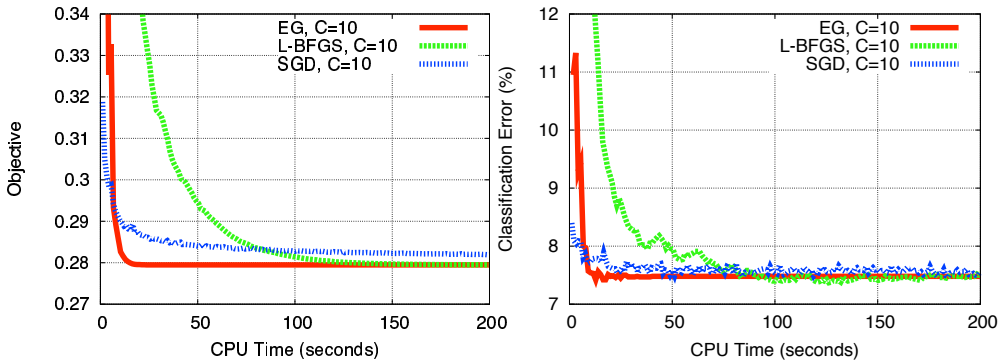


Figure 14: Timing results on the MNIST task, comparing the EG algorithm to L-BFGS and SGD. All algorithms are trained on the log-linear objective function with $C = 10$. The left figure shows objective values and the right figure shows classification error (see Figure 12). The results roughly correspond to 200 effective iterations.

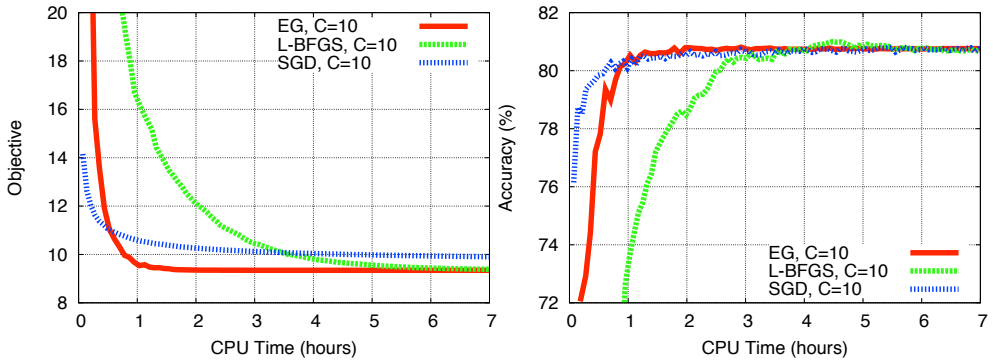


Figure 15: Timing results on the dependency-parsing task, comparing the EG algorithm to L-BFGS and SGD. All algorithms are trained on the log-linear objective function with $C = 10$. The left figure shows objective values and the right figure shows accuracy (see Figure 12). The results roughly correspond to 100 effective iterations.

SGD Update:

◁ *compute part-wise inner products*
1. $\mathbf{q} = \mathbf{0}$
2. **for** $r \in R(x_i)$
3. $\quad q_r = \mathbf{w}^t \cdot \mathbf{f}(x_i, r)$
4. **endfor**

◁ *compute marginals*
5. $\mathbf{m} = \text{MARGINALS}(\mathbf{q})$

◁ *apply gradient of regularizer*
6. $\mathbf{w}^{t+1} = (1 - \eta C)\mathbf{w}^t$

◁ *apply gradient of log-loss*
7. **for** $r \in R(x_i)$
8. $\quad$ **if** $r \in y_i$

◁ *empirical contribution*
9. $\quad\quad \mathbf{w}^{t+1} = \mathbf{w}^{t+1} + \eta \mathbf{f}(x_i, r)$
10. $\quad$ **endif**

◁ *expected contribution*
11. $\quad \mathbf{w}^{t+1} = \mathbf{w}^{t+1} - \eta m_r \mathbf{f}(x_i, r)$
12. **endfor**

Online EG Update:

◁ *compute part-wise inner products*
1. $\mathbf{q} = \mathbf{0}$
2. **for** $r \in R(x_i)$
3. $\quad q_r = \frac{1}{C}\mathbf{w}^t \cdot \mathbf{f}(x_i, r)$
4. **endfor**

◁ *update part-wise duals*
5. **for** $r \in R(x_i)$
6. $\quad s_{i,r}^{t+1} = (1 - \eta) s_{i,r}^t + \eta q_r$
7. **endfor**

◁ *compute marginals and partition function*
8. $(\mu_i^{t+1}, Z) = \text{MARGINALS}(\mathbf{s}_i^{t+1})$

◁ *compute new dual entropy*
9. $H_i^{t+1} = \log Z$
10. **for** $r \in R(x_i)$
11. $\quad H_i^{t+1} = H_i^{t+1} - \mu_{i,r}^{t+1} s_{i,r}^{t+1}$
12. **endfor**

◁ *update primals*
13. $\mathbf{w}^{t+1} = \mathbf{w}^t$
14. **for** $r \in R(x_i)$
15. $\quad \mathbf{w}^{t+1} = \mathbf{w}^{t+1} + (\mu_{i,r}^t - \mu_{i,r}^{t+1}) \mathbf{f}(x_i, r)$
16. **endfor**

◁ *compute change in dual objective*
17. $\delta = \frac{1}{2C}\|\mathbf{w}^{t+1}\|^2 - H_i^{t+1} - (\frac{1}{2C}\|\mathbf{w}^t\|^2 - H_i^t)$

Figure 16: Pseudocode for the updates performed in SGD and online EG for structured log-linear models (note that ◁ denotes a comment). In EG, we maintain dual vectors $\mathbf{s}_i^t$, marginals μ_i^t, entropy values H_i^t, and a vector $\mathbf{w}^t = \mathbf{w}(\mathbf{u}^t)$. Note that line-search techniques can be implemented based on the δ value computed in line 17 of the EG update. A vector scaling operation is required in line 6 of SGD, and vector norm operations are required in line 17 of EG; these can be performed in $O(1)$ time using an appropriate representation (e.g., see Shalev-Shwartz et al., 2007).

References

J. Baker. Trainable grammars for speech recognition. In J.J. Wolf and D.H. Klatt, editors, *Proceedings of the 97th meeting of the Acoustical Society of America*, pages 547–550. Acoustical Society of America, New York, NY, 1979.

P. L. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for large–margin structured classification. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 113–120, Cambridge, MA, 2005. MIT Press.

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31:167–175, 2003.

L.M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

S. Buchholz and E. Marsi. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning*, pages 149–164, New York City, 2006. Association for Computational Linguistics.

R.H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

Y. Censor and S.A. Zenios. *Parallel Optimization*. Oxford University Press, 1997.

M. Civit and M. Antònia Martí. Design principles for a Spanish treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 61–77, 2002.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

T.M. Cover and J.A Thomas. *Elements of Information Theory*. Wiley, 1991.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

N. Cristianini, C. Campbell, and J. Shawe-Taylor. Multiplicative updatings for support-vector learning. Technical report, NC-TR-98-016, Neuro COLT, Royal Holloway College, 1998.

A. Globerson, T. Koo, X. Carreras, and M. Collins. Exponentiated gradient algorithms for log-linear structured prediction. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 305–312. ACM Press, New York, NY, 2007.

D. Hush, P. Kelly, C. Scovel, and I. Steinwart. QP algorithms with guaranteed accuracy and run time for support vector machines. *Journal of Machine Learning Research*, 7:733–769, 2006.

T. Jaakkola and D. Haussler. Probabilistic kernel regression models. In D. Heckerman and J. Whittaker, editors, *Proceedings of 7th Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann, San Francisco, CA, 1999.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM Press, New York, NY, 2006.

S.S. Keerthi, K.B. Duan, S.K. Shevade, and A. N. Poo. A fast dual algorithm for kernel logistic regression. *Machine Learning*, 61:151–165, 2005.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

J. Kivinen and M. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

K. Koh, S.J. Kim, and S. Boyd. An interior point method for large scale l_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

T. Koo, A. Globerson, X. Carreras, and M. Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 141–150. Association for Computational Linguistics, 2007.

J. Lafferty, A. McCallum, and F. Pereira. Conditonal random fields: Probabilistic models for segmenting and labeling sequence data. In C.E. Brodley and A.P. Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, San Francisco, CA, 2001. Morgan Kaufmann.

G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 447–454. MIT Press, Cambridge, MA, 2002.

Y. LeCun, L. Bottou, Y. Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

N. List, D. Hush, C. Scovel, and I. Steinwart. Gaps in support vector optimization. In *Proceedings of the 20th Conference on Learning Theory*, pages 336–348, 2007.

R. McDonald, K. Crammer, and F. Pereira. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 91–98. Association for Computational Linguistics, 2005.

R. Memisevic. Dual optimization of conditional probability models. Technical report, University of Toronto, 2006.

T. Minka. A comparison of numerical optimizers for logistic regression. Technical report, Carnegie Mellon University, 2003.

M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 41–64. MIT Press, 1998.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141. Association for Computational Linguistics, 2003.

F. Sha, Y. Lin, L.K. Saul, and D.D. Lee. Multiplicative updates for nonnegative quadratic programming. *Neural Computation*, 19(8):2004–2031, 2007.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning*, pages 807–814. ACM Press, New York, NY, 2007.

B. Taskar, C. Guestrin, and D. Koller. Max margin Markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, Cambridge, MA, 2004a.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, 2004b.

B. Taskar, S. Lacoste-Julien, and M. Jordan. Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, pages 1627–1653, 2006.

C.H. Teo, Q. Le, A. Smola, and S.V.N. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM Press, New York, NY, USA, 2007.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In C.E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning*, pages 823–830. ACM, New York, NY, 2004.

S.V. N. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In W.W. Cohen and A. Moore, editors, *Proceedings of the 23rd International Conference on Machine Learning*, pages 969–976. ACM Press, New York, NY, 2006.

T. Zhang. On the dual formulation of regularized linear systems with convex risks. *Machine Learning*, 46:91–129, 2002.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1081–1088. MIT Press, Cambridge, MA, 2001.

Journal of Machine Learning Research 9 (2008) 1823-1840 Submitted 11/06; Revised 8/07; Published 8/08

Classification with a Reject Option using a Hinge Loss

Peter L. Bartlett BARTLETT@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720-1776, USA

Marten H. Wegkamp WEGKAMP@STAT.FSU.EDU
Department of Statistics
Florida State University
Tallahassee, FL 32306-4330, USA

Editor: John Shawe-Taylor

Abstract

We consider the problem of binary classification where the classifier can, for a particular cost, choose not to classify an observation. Just as in the conventional classification problem, minimization of the sample average of the cost is a difficult optimization problem. As an alternative, we propose the optimization of a certain convex loss function ϕ, analogous to the hinge loss used in support vector machines (SVMs). Its convexity ensures that the sample average of this surrogate loss can be efficiently minimized. We study its statistical properties. We show that minimizing the expected surrogate loss—the ϕ-risk—also minimizes the risk. We also study the rate at which the ϕ-risk approaches its minimum value. We show that fast rates are possible when the conditional probability $\mathbb{P}(Y=1|X)$ is unlikely to be close to certain critical values.

Keywords: Bayes classifiers, classification, convex surrogate loss, empirical risk minimization, hinge loss, large margin classifiers, margin condition, reject option, support vector machines

1. Introduction

The aim of binary classification is to classify observations that take values in an arbitrary feature space $\mathcal{X}$ into one of two classes, labeled -1 or $+1$. A *discriminant function* $f : \mathcal{X} \to \mathbb{R}$ yields a classifier $\mathrm{sgn}(f(x)) \in \{-1,+1\}$ that represents our guess of the label Y of a future observation X and we err if the margin $y \cdot f(x) < 0$. The Bayes discriminant function

$$\mathbb{P}\{Y=1|X=x\} - \mathbb{P}\{Y=-1|X=x\}$$

minimizes the probability of misclassification $\mathbb{P}\{Y f(X) < 0\}$. Observations x for which the conditional probability

$$\eta(x) = \mathbb{P}\{Y=+1|X=x\}$$

is close to $1/2$, are the most difficult to classify. In the extreme case where $\eta(x) = 1/2$, we may just as well toss a coin to make a decision. While it is our aim to classify the majority of future observations in an automatic way, it is often appropriate to instead report a warning for those observations that are hard to classify (the ones having conditional probability $\eta(x)$ near the value $1/2$). This motivates the introduction of a *reject option* for classifiers, by allowing for a third decision, ® (*reject*),

expressing doubt. For instance, in clinical trials it is important to be able to reject a tumor diagnostic classification since the consequences of misdiagnosis are severe and scientific expertise is required to make reliable determination. Although such classifiers are valuable in practice, few theoretical results are available in the statistical literature (Herbei and Wegkamp, 2006; Ripley, 1996). In the engineering community on the other hand this option is more common and empirically shown to effectively reduce the misclassification rate (Chow, 1970; Fumera and Roli, 2002, 2004; Fumera et al., 2000; Golfarelli et al., 1997; Györfi et al., 1978; Hansen et al., 1997; Landgrebe et al., 2006).

We propose to incorporate the reject option into our classification scheme by using a threshold value $0 \le \delta < 1$ as follows. Given a discriminant function $f : \mathcal{X} \to \mathbb{R}$, we report $\mathrm{sgn}(f(x))) \in \{-1,1\}$ if $|f(x)| > \delta$, but we withhold decision if $|f(x)| \le \delta$ and report ®. In this note, we assume that the cost of making a wrong decision is 1 and the cost of using the reject option is $d > 0$. The appropriate risk function is then

$$L_{d,\delta}(f) = \mathbb{E}\ell_d(Yf(X)) = \mathbb{P}\{Yf(X) < -\delta\} + d\mathbb{P}\{|Yf(X)| \le \delta\} \tag{1}$$

for the discontinuous loss

$$\ell_{d,\delta}(z) = \begin{cases} 1 & \text{if } z < -\delta, \\ d & \text{if } |z| \le \delta, \\ 0 & \text{otherwise.} \end{cases}$$

The classifier associated with the discriminant function $f_d^*(x)$ that minimizes the risk $L_{d,\delta}(f)$ assigns $-1, 1$ or ® depending on which of $\eta(x)$, $1-\eta(x)$ or d is smallest. Since we never reject if $d > 1/2$, we restrict ourselves to the cases $0 \le d \le 1/2$. The generalized Bayes discriminant function $f_d^*(x)$ is then

$$f_d^*(x) = \begin{cases} -1 & \text{if } \eta(x) < d \\ 0 & \text{if } d \le \eta(x) \le 1-d \\ +1 & \text{if } \eta(x) > 1-d \end{cases} \tag{2}$$

with risk

$$L_d^* = L_{d,\delta}(f_d^*) = \mathbb{E}\min\{\eta(X), 1-\eta(X), d\}.$$

The case $(\delta, d) = (0, 1/2)$ reduces to the classical situation without the reject option. We emphasize that the rejection cost d should be known a priori. In a medical setting when determining whether a disease is present or absent, the reject option often leads to quantifiable costs for additional tests and perhaps in delays of treatment. The exact value of d will be dictated by such considerations. From the above we can also view d as an upper bound on the conditional probability of misclassification (given X) that is considered tolerable.

We postpone the discussion on the choice of the threshold δ until after Theorem 2.

Plug-in classification rules replace the regression function $\eta(x)$ by an estimate $\widehat{\eta}(x)$ in the formula for $f_d^*(x)$ above. It is shown by Herbei and Wegkamp (2006) that the rate of convergence of the risk (1) to the Bayes risk L_d^* of a general plug-in rule with reject option depends on how well $\widehat{\eta}(X)$ estimates $\eta(X)$ and on the behavior of $\eta(X)$ near the values d and $1-d$. This condition on $\eta(X)$ nicely generalizes the margin condition of Tsybakov (2004) from the classical setting ($d = 1/2$) to our more general framework ($0 \le d \le 1/2$). The same paper derives oracle inequalities for the excess risk $L_{d,\delta}(\widehat{f}) - L_d^*$ of the (naive) empirical risk minimizer $\widehat{f}$ of $\sum_{i=1}^n \ell_{d,\delta}(Y_i f(X_i))$ based on n

independent observations (X_i, Y_i), over a class of discriminant functions $\mathcal{F}$. The results are in line with recent theoretical developments (Boucheron et al., 2006, 2005; Massart, 2007) of standard binary classification ($d = 1/2$). Despite its attractive theoretical properties, the naive empirical risk minimization method is often hard to implement. This paper addresses this pitfall by considering a convex surrogate for the loss function akin to the hinge loss that is used in SVMs. In the engineering literature, there are recently encouraging empirical results on SVMs with a reject option (Bounsiar et al., 2006; Fumera et al., 2003; Fumera and Roli, 2002; Tortorella, 2004).

The next section introduces a piecewise linear loss function $\phi_d(x)$ that generalizes the hinge loss function $\max\{0, 1-x\}$ in that it allows for the reject option and $\phi_d(x) = \max\{0, 1-x\}$ for $d = 1/2$. We prove that f_d^* in (2) also minimizes the risk associated with this new loss and that the excess risk $L_{d,\delta} - L_d^*$ can be bounded by $2d$ times the excess risk based on the piecewise linear loss ϕ_d if $\delta = 1/2$. Thus classifiers with small excess ϕ_d-risk automatically have small excess classification risk, providing theoretical justification of the more computationally appealing method.

In Section 3, we illustrate the computational convenience of the new loss, showing that the SVM classifier with reject option can be obtained by solving a standard convex optimization problem.

Finally, in Section 4, we show that fast rates (for instance, faster than $n^{-1/2}$) of the SVM classifier with reject option are possible under the same noise conditions on $\eta(X)$ used by Herbei and Wegkamp (2006). As a side effect, for the standard SVM (the special case of $d = 1/2$), our results imply fast rates without an assumption that $\eta(X)$ is unlikely to be near 0 and 1, a technical condition that has been imposed in the literature for that case (Blanchard et al., 2008; Tarigan and van de Geer, 2006).

2. Generalized Hinge Loss

Instead of the discontinuous loss $\ell_{d,\delta}$, we consider the convex surrogate loss

$$\phi_d(z) = \begin{cases} 1 - az & \text{if } z < 0, \\ 1 - z & \text{if } 0 \le z < 1, \\ 0 & \text{otherwise} \end{cases}$$

where $a = (1-d)/d \ge 1$. The next result states that the minimizer of the expectation of the discrete loss $\ell_{d,\delta}(z)$ and the convex loss $\phi_d(z)$ remains the same.

Proposition 1 *The Bayes discriminant function (2) minimizes the risk*

$$L_{\phi_d}(f) = \mathbb{E}\phi_d(Y f(X))$$

over all measurable $f : \mathcal{X} \to \mathbb{R}$. *Furthermore,*

$$dL_{\phi_d}(f_d^*) = L_{d,\delta}(f_d^*).$$

Proof Observe that

$$L_{\phi_d}(f) = \mathbb{E}\eta(X)\phi_d(f(X)) + \mathbb{E}(1-\eta(X))\phi_d(-f(X)).$$

Hence, for

$$r_{\eta,\phi_d}(z) = \eta\phi_d(z) + (1-\eta)\phi_d(-z) \tag{3}$$

it suffices to show that

$$z^* = \begin{cases} -1 & \text{if } \eta < 1/(1+a), \\ 0 & \text{if } 1/(1+a) \le \eta \le a/(1+a), \\ 1 & \text{if } \eta > a/(1+a) \end{cases}$$

minimizes $r_{\eta,\phi_d}(z)$. The function $r_{\eta,\phi_d}(z)$ can be written as

$$r_{\eta,\phi_d}(z) = \begin{cases} \eta - a\eta z & \text{if } z \le -1, \\ 1 + z(1-(1+a)\eta) & \text{if } -1 \le z \le 0, \\ 1 + z(-\eta + a(1-\eta)) & \text{if } 0 \le z \le 1, \\ z(a(1-\eta)) + (1-\eta) & \text{if } z \ge 1 \end{cases}$$

and it is now a simple exercise to verify that z^* indeed minimizes $r_{\eta,\phi_d}(z)$. Finally, since $L_{\phi_d}(f) = \mathbb{E} r_{\eta,\phi_d}(f(X))$ and

$$\begin{aligned} & \inf_z \eta\phi_d(z) + (1-\eta)\phi_d(-z) \\ = \;& \eta\phi_d(z^*) + (1-\eta)\phi_d(z^*) \\ = \;& \frac{\eta}{d}\mathbf{1}[\eta < d] + \mathbf{1}[d \le \eta \le 1-d] + \frac{1-\eta}{d}\mathbf{1}[\eta > 1-d], \end{aligned}$$

where $\mathbf{1}[A]$ denotes the indicator function of a set A, we find that

$$d L_{\phi_d}(f_d^*) = \mathbb{E}[\min(\eta(X), 1-\eta(X), d)] = L_d^*.$$

and the second claim follows as well. ■

We see that $\phi_d(z) \ge \ell_{d,\delta}(z)$ for all $z \in \mathbb{R}$ as long as $0 \le \delta \le 1-d$. Since this pointwise relation remains preserved under taking expected values, we immediately obtain $L_{d,\delta}(f) \le L_{\phi_d}(f)$. The following comparison theorem shows that a relation like this holds not only for the risks, but for the excess risks as well.

Theorem 2 *Let $0 \le d < 1/2$ and a measurable function f be fixed. For all $0 < \delta \le 1/2$, we have*

$$L_{d,\delta}(f) - L_d^* \le \frac{d}{\delta}\left(L_{\phi_d}(f) - L_{\phi_d}^*\right),$$

where $L_{\phi_d}^* = L_{\phi_d}(f_d^*)$. For $1/2 \le \delta \le 1-d$, we have

$$L_{d,\delta}(f) - L_d^* \le L_{\phi_d}(f) - L_{\phi_d}^*.$$

Finally, for $(\delta, d) = (0, 1/2)$, we have

$$L(f) - L^* \le L_\phi(f) - L_\phi^*, \tag{4}$$

where $L(f) := \mathbb{P}\{Y f(X) < 0\}$, $L^* := \mathbb{E}\min(\eta(X), 1-\eta(X))$ and $\phi(x) = \max\{0, 1-x\}$.

Remark 3 *The optimal multiplicative constant (d/δ or 1 depending on the value of δ) in front of the ϕ_d-excess risk is achieved at $\delta = 1/2$. For this choice, Theorem 2 states that*

$$L_{d,1/2}(f) - L_d^* \leq 2d\left(L_{\phi_d}(f) - L_{\phi_d}^*\right).$$

For all $d \leq \delta \leq 1-d$, the multiplicative constant in front of the ϕ_d-excess risk does not exceed 1. The choice $\delta = 1/2$ with the smallest constant $2d < 1$ is right in the middle of the interval $[d, 1-d]$. The choice $\delta = 1-d$ corresponds to the largest value of δ for which the piecewise constant function $\ell_{d,\delta}(z)$ is still majorized by the convex surrogate $\phi_d(z)$. For $\delta = d$ we will reject less frequently than for $\delta = 1-d$ and $\delta = 1/2$ can be seen as a compromise among these two extreme cases.

Inequality (4) is due to Zhang (2004).

Before we prove the theorem, we need an intermediate result. We define the functions

$$\xi(\eta) = \eta \mathbf{1}[\eta < d] + d\mathbf{1}[d \leq \eta \leq 1-d] + (1-\eta)\mathbf{1}[\eta > 1-d]$$

and

$$\begin{aligned} H(\eta) &= \inf_z \eta\phi_d(z) + (1-\eta)\phi_d(-z) \\ &= \frac{\eta}{d}\mathbf{1}[\eta < d] + \mathbf{1}[d \leq \eta \leq 1-d] + \frac{1-\eta}{d}\mathbf{1}[\eta > 1-d]. \end{aligned}$$

(We suppress their dependence on d in our notation.) Their expectations are $L_d^* = \mathbb{E}\xi(\eta(X))$ and $L_{\phi_d}^* = \mathbb{E}H(\eta(X))$, respectively. Furthermore, we define

$$\begin{aligned} H_{-1}(\eta) &= \inf_{z<-\delta}(\eta\phi_d(z) + (1-\eta)\phi_d(-z)), \\ H_{\textcircled{R}}(\eta) &= \inf_{|z|\leq\delta}(\eta\phi_d(z) + (1-\eta)\phi_d(-z)), \\ H_1(\eta) &= \inf_{z>\delta}(\eta\phi_d(z) + (1-\eta)\phi_d(-z)); \\ \xi_{-1}(\eta) &= \eta - \xi(\eta), \\ \xi_{\textcircled{R}}(\eta) &= d - \xi(\eta), \\ \xi_1(\eta) &= 1 - \eta - \xi(\eta). \end{aligned}$$

Proposition 4 *Let $0 \leq d < 1/2$.*

If $0 < \delta \leq 1/2$, then, for $b \in \{-1, 1, \textcircled{R}\}$,

$$\xi_b(\eta) \leq \frac{\delta}{d}\{H_b(\eta) - H(\eta)\}.$$

If $d \leq \delta \leq 1-d$, then, for $b \in \{-1, 1, \textcircled{R}\}$,

$$\xi_b(\eta) \leq H_b(\eta) - H(\eta).$$

If $(\delta, d) = (0, 1/2)$, then, for $b \in \{-1, 1, \textcircled{R}\}$,

$$\xi_b(\eta) \leq H_b(\eta) - H(\eta).$$

The proof is in the appendix.

Proof [Proof of Theorem 2] Recall that $L_{d,\delta}(f) = \mathrm{P}(\eta \mathbf{1}[f < -\delta] + d\mathbf{1}[-\delta \le f \le \delta] + (1-\eta)\mathbf{1}[f > \delta])$ and $L_{\phi_d}(f) = \mathrm{P}r_{\eta,\phi_d}(f)$ with r_{η,ϕ_d} defined in the proof of Proposition 1. Here P is the probability measure of X and $\mathrm{P}g = \int g d\mathrm{P}$ for any P-integrable g. Assume $0 < \delta \le 1/2$ and $0 \le d < 1/2$. Define $\psi(x) = x\delta/d$. By linearity of ψ, we have for any measurable function f,

$$\begin{aligned}\psi(L_{d,\delta}(f) - L_d^*) &= \mathrm{P}\left(\mathbf{1}[f < -\delta]\,\psi(\xi_{-1}(\eta)) + \mathbf{1}[-\delta \le f \le \delta]\,\psi(\xi_{\circledR}(\eta))\right. \\ &\qquad \left. + \mathbf{1}[f > \delta]\,\psi(\xi_1(\eta))\right).\end{aligned}$$

Invoke now Proposition 4 to deduce

$$\begin{aligned}\psi(L_{d,\delta}(f) - L_d^*) &\le \mathrm{P}\left(\mathbf{1}[f < -\delta]\,[H_{-1}(\eta) - H(\eta)] + \mathbf{1}[-\delta \le f \le \delta]\,[H_{\circledR}(\eta) - H(\eta)]\right. \\ &\qquad \left. + \mathbf{1}[f > \delta]\,[H_1(\eta) - H(\eta)]\right) \\ &\le \mathrm{P}\left\{r_{\eta,\phi_d}(f) - H(\eta)\right\}\end{aligned}$$

and conclude the proof by observing that the term on the right of the previous inequality equals $L_{\phi_d}(f) - L_{\phi_d}^*$.

For the case $(\delta, d) = (0, 1/2)$ and the case (δ, d) with $d \le \delta \le 1 - d$ and $0 \le d < 1/2$, take $\psi(x) = x$. ■

3. SVM Classifiers with Reject Option

In this section, we consider an SVM-like classifier for classification with a reject option, and show that it can be obtained by solving a quadratically constrained quadratic program (QCQP).

Let $K : \mathcal{X}^2 \to \mathbb{R}$ be the kernel of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$, and let $\|f\|$ be the norm of f in $\mathcal{H}$. The SVM classifier with reject option is the minimizer of the empirical ϕ_d-risk subject to a constraint on the RKHS norm.[1] The following theorem shows that this classifier is the solution to a QCQP, that is, it is the minimizer of a convex quadratic criterion on a convex subset of Euclidean space defined by quadratic inequalities. Thus, the classifier can be found efficiently using general-purpose algorithms.

Theorem 5 *For any $x_1, \ldots, x_n \in \mathcal{X}$ and $y_1, \ldots, y_n \in \{-1, 1\}$, let $\widehat{f} \in \mathcal{H}$ be the solution to*

$$\begin{aligned}&\textit{minimize} && f \mapsto \sum_{i=1}^n \phi_d(y_i f(x_i)) \\ &\textit{such that} && \|f\|^2 \le r^2,\end{aligned}$$

where $r > 0$. Then we can represent $\widehat{f}$ as the finite sum

$$\widehat{f}(x) = \sum_{i=1}^n \widehat{\alpha}_i K(x_i, x),$$

1. Notice that we parameterize the optimization problem in terms of the constraint on the RKHS norm, rather than in terms of its Lagrange multiplier, which is more standard. The regularization path—the set of solutions to these problems as the parameter of the optimization problem varies—is identical.

where $\widehat{\alpha}_1, \ldots, \widehat{\alpha}_n$ is the solution to the following QCQP.

$$\begin{aligned}
\min_{\alpha_i, \xi_i, \gamma_i} \quad & \frac{1}{n} \sum_{i=1}^{n} \left(\xi_i + \frac{1-2d}{d} \gamma_i \right) \\
\textit{such that} \quad & \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \leq r^2 \\
& \xi_i \geq 0, \qquad \gamma_i \geq 0, \\
& \xi_i \geq 1 - y_i \sum_{j=1}^{n} \alpha_j K(x_i, x_j), \\
& \gamma_i \geq -y_i \sum_{j=1}^{n} \alpha_j K(x_i, x_j) \qquad\qquad \textit{for } i = 1, \ldots, n.
\end{aligned}$$

Proof The fact that $\widehat{f}$ can be represented as a finite sum over the kernel basis functions is a standard argument (Kimeldorf and Wahba, 1971; Cox and O'Sullivan, 1990). It follows from Pythagoras' theorem in Hilbert space: the squared RKHS norm can be split into the squared norm of the component in the space spanned by the kernel basis functions $x \mapsto K(x_i, x)$ and that of the component in the orthogonal subspace. Since the cost function depends on f only at the points x_i, and the reproducing property $f(x_i) = \langle K(x_i, \cdot), f \rangle$ shows that these values depend only on the component of f in the space spanned by the kernel basis functions, the orthogonal subspace only makes the constraint harder to satisfy, but does not affect the cost function. Thus, a minimizing $\widehat{f}$ can be represented in terms of the solution $\widehat{\alpha}$ to the minimization

$$\begin{aligned}
\min_{\alpha_1, \ldots, \alpha_n} \quad & \frac{1}{n} \sum_{i=1}^{n} \phi_d \left(y_i \sum_{j=1}^{n} \alpha_j K(x_i, x_j) \right) \\
\text{such that} \quad & \sum_{1 \leq i,j \leq n} \sum \alpha_i \alpha_j K(x_i, x_j) \leq r^2.
\end{aligned}$$

But then it is easy to see that we can decompose ϕ_d as

$$\phi_d(\beta) = \max\{0, 1-\beta\} + \frac{1-2d}{d} \max\{0, -\beta\}.$$

Parameterizing ϕ_d using the slack variables

$$\xi_i = \max\{0, 1 - y_i f(x_i)\}, \qquad \gamma_i = \max\{0, -y_i f(x_i)\}$$

gives the QCQP. ■

4. Tsybakov's Margin Condition, Bernstein Classes, and Fast Rates

In this section, we consider methods that choose the function $\widehat{f}$ from some class $\mathcal{F}$ so as to minimize the empirical ϕ_d-risk

$$\widehat{L}_{\phi_d}(f) = \frac{1}{n} \sum_{i=1}^{n} \phi_d(Y_i f(X_i)).$$

For instance, to analyze the SVM classifier with reject option, we could consider classes $\mathcal{F}_n = \{f \in \mathcal{H} : \|f\| \leq c_n\}$ for some sequence of constants c_n. We are interested in bounds on the excess ϕ_d-risk, that is, the difference between the ϕ_d-risk of $\widehat{f}$ and the minimal ϕ_d-risk over all measurable functions, of the form

$$\mathbb{E}L_{\phi_d}(\widehat{f}) - L^*_{\phi_d} \leq 2 \inf_{f \in \mathcal{F}} \left(L_{\phi_d}(f) - L^*_{\phi_d}\right) + \varepsilon_n.$$

Such bounds can be combined with an assumption on the rate of decrease of the approximation error $\inf_{f \in \mathcal{F}_n} \left(L_{\phi_d}(f) - L^*_{\phi_d}\right)$ for a sequence of classes $\mathcal{F}_n$ used by a method of sieves, and thus provide bounds on the rate of convergence of risk $L_{d,\delta}(\widehat{f})$ to the optimal Bayes risk L^*_d.

For many binary classification methods (including empirical risk minimization, plug-in estimates, and minimization of the sample average of a suitable convex loss), the estimation error term ε_n approaches zero at a faster rate when the conditional probability $\eta(X)$ is unlikely to be close to the critical value of $1/2$ (Audibert and Tsybakov, 2007; Bartlett et al., 2006; Blanchard et al., 2008; Steinwart and Scovel, 2007; Tarigan and van de Geer, 2006; Tsybakov, 2004). For plug-in rules, Herbei and Wegkamp (2006) showed an analogous result for classification with a reject option, where the corresponding condition concerns the probability that $\eta(X)$ is close to the critical values of d and $1-d$. In this section, we prove a bound on the excess ϕ_d-risk of $\widehat{f}$ that converges rapidly when a condition of this kind applies. We begin with a precise statement of the condition. For $d = 1/2$, it is equivalent to the margin condition of Tsybakov (2004).

Definition 6 *We say that η satisfies the margin condition at d with exponent $\alpha > 0$ if there is a $c \geq 1$ such that for all $t > 0$,*

$$\mathbb{P}\{|\eta(X) - d| \leq t\} \leq ct^\alpha \text{ and } \mathbb{P}\{|\eta(X) - (1-d)| \leq t\} \leq ct^\alpha.$$

The reason that conditions of this kind allow fast rates is related to the variance of the excess ϕ_d-loss,

$$g_f(x,y) = \phi_d(yf(x)) - \phi_d(yf^*_d(x)),$$

where f^*_d minimizes the ϕ_d-risk. Notice that the expectation of g_f is precisely the excess risk of f, $\mathbb{E}g_f(X,Y) = L_{\phi_d}(f) - L^*_{\phi_d}$. We will show that when η satisfies the margin condition at d with exponent α, the variance of each g_f is bounded in terms of its expectation, and thus approaches zero as the ϕ-risk of f approaches the minimal value. Classes for which this occurs are called Bernstein classes.

Definition 7 *We say that $\mathcal{G} \subset L_2(\mathrm{P})$ is a (β, B)-Bernstein class with respect to the probability measure P $(0 < \beta \leq 1, B \geq 1)$ if every $g \in \mathcal{G}$ satisfies*

$$\mathrm{P}g^2 \leq B(\mathrm{P}g)^\beta.$$

We say that $\mathcal{G}$ has a Bernstein exponent β with respect to P if there exists a constant B for which $\mathcal{G}$ is a (β, B)-Bernstein class.

Lemma 8 *If η satisfies the margin condition at d with exponent α, then for any class $\mathcal{F}$ of measurable uniformly bounded functions, the class $\mathcal{G} = \{g_f : f \in \mathcal{F}\}$ has a Bernstein exponent $\beta = \alpha/(1+\alpha)$.*

The result relies on the following two lemmas. The first shows that the excess ϕ_d-risk is at least linear in a certain pseudo-norm of the difference between f and f_d^*. It is similar to the $L_1(\mathrm{P})$ norm, but it penalizes f less for large excursions that have little impact on the ϕ_d-risk. For example, if $\eta(x) = 1$, then the conditional ϕ_d-risk is zero even if $f(x)$ takes a large positive value. For $\eta \in [0,1]$, define

$$\rho_\eta(f, f_d^*) = \begin{cases} \eta|f - f_d^*| & \text{if } \eta < d \text{ and } f < -1, \\ (1-\eta)|f - f_d^*| & \text{if } \eta > 1-d \text{ and } f > 1, \\ |f - f_d^*| & \text{otherwise,} \end{cases}$$

and recall the definition of the conditional ϕ_d-risk in (3).

Lemma 9 *For* $\eta \in [0,1]$,

$$d\left(r_{\eta,\phi_d}(f) - r_{\eta,\phi_d}(f_d^*)\right) \geq (|\eta - d| \wedge |\eta - (1-d)|)\,\rho_\eta(f, f_d^*).$$

Proof Since r_{η,ϕ_d} is convex,

$$r_{\eta,\phi_d}(f) \geq r_{\eta,\phi_d}(f_d^*) + g(f - f_d^*)$$

for any g in the subgradient of $r_{\eta,\phi_d}(f)$ at f_d^*. In our case, r_{η,ϕ_d} is piecewise linear, with four pieces, and the subgradients include

$$\begin{array}{ll} \eta\frac{1-d}{d} & \text{at } f_d^* = -1, \\ |\eta - d|\frac{1}{d} & \text{at } f_d^* = -1, 0, \\ |1 - \eta - d|\frac{1}{d} & \text{at } f_d^* = 0, 1, \\ (1-\eta)\frac{1-d}{d} & \text{at } f_d^* = 1. \end{array}$$

Thus, we have

$$\begin{aligned} & d(r_{\eta,\phi_d}(f) - r_{\eta,\phi_d}(f_d^*)) \\ & \geq \begin{cases} \eta(1-d)|f - f_d^*| & \text{if } \eta < d \text{ and } f < -1, \\ |\eta - d||f - f_d^*| & \text{if } \eta < d \text{ and } f > -1, \\ (|\eta - d| \wedge |1 - \eta - d|)\,|f - f_d^*| & \text{if } d \leq \eta \leq 1-d, \\ |1 - \eta - d||f - f_d^*| & \text{if } \eta > 1-d \text{ and } f < 1, \\ (1-\eta)(1-d)|f - f_d^*| & \text{if } \eta > 1-d,\ f > 1. \end{cases} \\ & = \begin{cases} (1-d)\rho_\eta(f, f_d^*) & \text{if } \eta < d \text{ and } f < -1, \\ |\eta - d|\rho_\eta(f, f_d^*) & \text{if } \eta < d \text{ and } f > -1, \\ (|\eta - d| \wedge |1 - \eta - d|)\,\rho_\eta(f, f_d^*) & \text{if } d \leq \eta \leq 1-d, \\ |1 - \eta - d|\rho_\eta(f, f_d^*) & \text{if } \eta > 1-d \text{ and } f < 1, \\ (1-d)\rho_\eta(f, f_d^*) & \text{if } \eta > 1-d,\ f > 1. \end{cases} \\ & \geq (|\eta - d| \wedge |1 - \eta - d|)\,\rho_\eta(f, f_d^*). \end{aligned}$$

■

We shall also use the following inequalities.

Lemma 10 *If $\|f\|_\infty = B$, then for $\eta \in [0,1]$,*

$$\rho_\eta(f, f_d^*) \le |f - f_d^*|,$$

and

$$\eta\,|\phi_d(f) - \phi_d(f_d^*)|^2 + (1-\eta)\,|\phi_d(-f) - \phi_d(-f_d^*)|^2 \le \left(\frac{1-d}{d}\right)^2 (B+1)\rho_\eta(f, f_d^*).$$

Proof The first inequality is immediate from the definition of ρ_η. To see the second, use the fact that ϕ_d is flat to the right of 1 to notice that

$$\begin{aligned}
&\eta\,|\phi_d(f) - \phi_d(f_d^*)|^2 + (1-\eta)\,|\phi_d(-f) - \phi_d(-f_d^*)|^2 \\
&= \begin{cases} \eta\left|\phi_d(f) - \phi_d(f_d^*)\right|^2 & \text{if } \eta < d \text{ and } f < -1, \\ (1-\eta)\left|\phi_d(-f) - \phi_d(-f_d^*)\right|^2 & \text{if } \eta > 1-d \text{ and } f > 1. \end{cases}
\end{aligned}$$

Since ϕ_d has Lipschitz constant $a = (1-d)/d$, this implies

$$\begin{aligned}
&\eta\,|\phi_d(f) - \phi_d(f_d^*)|^2 + (1-\eta)\,|\phi_d(-f) - \phi_d(-f_d^*)|^2 \\
&\le \begin{cases} \eta a^2 |f - f_d^*|^2 & \text{if } \eta < d \text{ and } f < -1, \\ (1-\eta) a^2 |f - f_d^*|^2 & \text{if } \eta > 1-d \text{ and } f > 1, \\ a^2 |f - f_d^*|^2 & \text{otherwise} \end{cases} \\
&\le a^2(1+B)\rho_\eta(f, f_d^*),
\end{aligned}$$

where the last inequality uses the fact that $|f - f_d^*| \le B+1$. ■

Proof [Proof of Lemma 8] By Lemma 9, we have

$$L_{\phi_d}(f) - L_{\phi_d}^* \ge d^{-1}\mathbb{E}\rho_\eta(f, f_d^*)\left(|\eta - (1-d)|I_{E_-} + |\eta - d|I_{E_+}\right),$$

with

$$E_- = \{|\eta - (1-d)| \le |\eta - d|\}, \qquad E_+ = \{|\eta - (1-d)| > |\eta - d|\}.$$

Using the assumption on η, there is an $A \ge 1$ such that for all $t > 0$

$$\mathbb{P}\{|\eta(X) - d| \le t\} \le At^\alpha \text{ and } \mathbb{P}\{\eta(X) - (1-d)| \le t\} \le At^\alpha.$$

Thus, for any set E,

$$\begin{aligned}
\mathrm{P}\rho_\eta(f, f_d^*)|\eta - (1-d)|I_E &\ge t\mathrm{P}\rho_\eta(f, f_d^*)I_{\{|\eta-(1-d)|\ge t\}}I_E \\
&= t\mathrm{P}\rho_\eta(f, f_d^*)I_E - t\mathrm{P}\rho_\eta(f, f_d^*)I_{\{|\eta-(1-d)|<t\}}I_E \\
&\ge t\{\mathrm{P}\rho_\eta(f, f_d^*)I_E - (B+1)At^\alpha\},
\end{aligned}$$

where B is such that $|f| \le B$ and hence $\rho_\eta(f, f_d^*) \le |f - f_d^*| \le B+1$. Similarly,

$$\mathrm{P}\rho_\eta(f, f_d^*)|\eta - d|I_E \ge t\{\mathrm{P}\rho_\eta(f, f_d^*)I_E - (B+1)At^\alpha\},$$

and we obtain

$$\begin{aligned} L_{\phi_d}(f) - L^*_{\phi_d} &\geq d^{-1}t\left(\mathrm{P}\rho_\eta(f, f_d^*) I_{E_+ \cup E_-} - 2(B+1)At^\alpha\right) \\ &= d^{-1}t\left(\mathrm{P}\rho_\eta(f, f_d^*) - 2(B+1)At^\alpha\right). \end{aligned}$$

Choose

$$t = \left(\frac{\mathrm{P}\rho_\eta(f, f_d^*)}{4(B+1)A}\right)^{1/\alpha},$$

in the expression above, and we obtain

$$\mathbb{E}g_f(X,Y) = L_{\phi_d}(f) - L^*_{\phi_d} \geq \frac{1}{2d(4(B+1)A)^{1/\alpha}} \left(\mathrm{P}\rho_\eta(f, f_d^*)\right)^{(1+\alpha)/\alpha},$$

and so

$$\mathrm{P}\rho_\eta(f, f_d^*) \leq \left\{2d(4(B+1)A)^{1/\alpha}\right\}^{\alpha/(\alpha+1)} \left\{\mathbb{E}g_f(X,Y)\right\}^{\alpha/(1+\alpha)}.$$

In addition, by Lemma 10,

$$\begin{aligned} \mathbb{E}\{g_f(X,Y)\}^2 &= \mathbb{E}\mathbb{E}[\{g_f(X,Y)\}^2|X] \\ &= \mathrm{P}\left(\eta|\phi_d(f) - \phi_d(f_d^*)|^2 + (1-\eta)|\phi_d(-f) - \phi_d(-f_d^*)|^2\right) \\ &\leq (B+1)\left(\frac{1-d}{d}\right)^2 \mathrm{P}\rho_\eta(f, f_d^*). \end{aligned}$$

Combining these two inequalities shows that

$$\mathbb{E}\{g_f(X,Y)\}^2 \leq (B+1)\left(\frac{1-d}{d}\right)^2 \left(2d(4A(B+1))^{1/\alpha}\right)^{\alpha/(\alpha+1)} \left(\mathbb{E}g_f(X,Y)\right)^{\alpha/(1+\alpha)}.$$

■

Remark 11 *Specialized to the case $(\delta, d) = (0, 1/2)$, we note that Lemma 8 removes unnecessary technical restrictions on $\eta(X)$ near 0 and 1, imposed by Blanchard et al. (2008) and Tarigan and van de Geer (2006). This is consistent with results of Steinwart and Scovel (2007) on SVMs with Gaussian kernels.*

Lemma 8 provides the main ingredient for establishing fast rates of minimizers $\widehat{f}_d$ of the empirical risk $\widehat{L}_{\phi_d}(f)$.

In the theorem, we use the notation $N(\varepsilon, L_\infty, \mathcal{F})$ to denote the ε-covering number of $\mathcal{F}$ in L_∞, that is, the smallest number of closed ε-balls in L_∞ needed to cover $\mathcal{F}$. The countability assumption means that measurability is not an issue. It can be replaced by other mild sufficient conditions.

Theorem 12 *If η satisfies the margin condition at d with exponent α, $\mathcal{F}$ is a countable class of functions $f : \mathcal{X} \to \mathbb{R}$ satisfying $\|f\|_\infty \leq B$, and $\mathcal{F}$ satisfies*

$$\log N(\varepsilon, L_\infty, \mathcal{F}) \leq C\varepsilon^{-p}$$

for all $\varepsilon > 0$ *and some* $0 \le p \le 2$*, then there exists a constant* C' *independent of* n*, such that*

$$\mathbb{E}L_{\phi_d}(\widehat{f}_d) - L^*_{\phi_d} \le 2 \inf_{f\in\mathcal{F}} \left(L_{\phi_d}(f) - L^*_{\phi_d}\right) + C' n^{-\frac{1+\alpha}{2+p+\alpha+p\alpha}},$$

where $\widehat{f}_d = \arg\min_{f\in\mathcal{F}} \widehat{L}_{\phi_d}(f)$.

Proof We use the notation $\mathrm{P}g_f = \mathbb{E}g_f(X,Y)$ and

$$\mathbb{P}_n g_f = \frac{1}{n}\sum_{i=1}^n g_f(X_i,Y_i).$$

By definition of $\widehat{f}_d$, we have

$$\begin{aligned} L_{\phi_d}(\widehat{f}_d) - L^*_{\phi_d} &= \mathrm{P}g_{\widehat{f}_d} \\ &= 2\mathbb{P}_n g_{\widehat{f}_d} + (\mathrm{P}-2\mathbb{P}_n)g_{\widehat{f}_d} \\ &\le 2\inf_{f\in\mathcal{F}} \mathbb{P}_n g_f + \sup_{f\in\mathcal{F}} (\mathrm{P}-2\mathbb{P}_n)g_f. \end{aligned}$$

Taking expected values on both sides, yields,

$$\mathbb{E}L_{\phi_d}(\widehat{f}_d) - L^*_{\phi_d} \le 2\inf_{f\in\mathcal{F}} \left(L_{\phi_d}(f) - L^*_{\phi_d}\right) + \mathbb{E}\left[\sup_{f\in\mathcal{F}} (\mathrm{P}-2\mathbb{P}_n)g_f\right].$$

Since $|g_f - g_{f'}| \le |f-f'|(1-d)/d$, it follows that

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}} (\mathrm{P}-2\mathbb{P}_n)g_f\right] \le \frac{1-d}{d}\varepsilon_n + \frac{1-d}{d}B\mathbb{P}\left\{\sup_{f\in\mathcal{F}_n} (\mathrm{P}-2\mathbb{P}_n)g_f \ge \varepsilon_n\right\},$$

where $\mathcal{F}_n$ is a minimal ε_n-covering net of $\mathcal{F}$ with

$$\varepsilon_n = Mn^{-(1+\alpha)/(2+p+\alpha+p\alpha)}$$

for some constant M to be selected later. The union bound and Bernstein's exponential inequality for the tail probability of sums of bounded random variables in conjunction with Lemma 8, yield

$$\begin{aligned} \mathbb{P}\left\{\sup_{f\in\mathcal{F}_n} (\mathrm{P}-2\mathbb{P}_n)g_f \ge \varepsilon_n\right\} &\le \sum_{f\in\mathcal{F}_n} \mathbb{P}\left\{(\mathrm{P}-\mathbb{P}_n)g_f \ge \frac{1}{2}(\mathrm{P}g_f + \varepsilon_n)\right\} \\ &\le |\mathcal{F}_n| \max_{f\in\mathcal{F}_n} \exp\left(-\frac{n}{8}\frac{(\varepsilon_n + \mathrm{P}g_f)^2}{\mathrm{P}g_f^2 + B(\varepsilon_n+\mathrm{P}g_f)/6}\right) \\ &\le \exp(C\varepsilon_n^{-p} - cn\varepsilon_n^{2-\beta}) \end{aligned}$$

with $0 \le \beta = \alpha/(1+\alpha) \le 1$ and some $c > 0$ independent of n. Conclude the proof by noting that

$$\exp(C\varepsilon_n^{-p} - cn\varepsilon_n^{2-\beta}) = \exp\left(-\frac{c}{2}n\varepsilon_n^{2-\beta}\right),$$

and by choosing the constant M in ε_n such that $C\varepsilon_n^{-p} = cn\varepsilon_n^{2-\beta}/2$ and $\exp(-n\varepsilon_n^{2-\beta}) = o(\varepsilon_n)$. ■

Remark 13 *The constant 2 in front of the minimal excess risk on the right could be made closer to* 1*, at the expense of increasing* C'*.*

Theorem 12 discusses minimizers of the empirical risk $\widehat{L}_{\phi_d}$ *over classes* $\mathcal{F}$ *of uniformly bounded functions. The analysis of SVMs that minimize* $\widehat{L}_{\phi_d}$ *plus a regularization term requires more work.*

Remark 14 *Consider for simplicity the case* $\mathcal{F}$ *is finite (*$p = 0$*). Then, if the margin condition holds for* $\alpha = +\infty$*, we obtain from the proof of Theorem 12 rates of convergence of order* $\log|\mathcal{F}|/n$*. If* $\alpha = 0$*, we in fact impose no restriction on* $\eta(X)$ *at all, and the rate equals* $(\log|\mathcal{F}|/n)^{1/2}$*.*

Remark 15 *The entropy condition is satisfied for many classes. For instance, Kolmogorov and Tichomirov (1961) prove the following result for Sobolev spaces with parameter* β*. Let* $\mathcal{X}$ *be a bounded, convex subset of* $\mathbb{R}^d$ *and for every* $k = (k_1, \ldots, k_d) \in \mathbb{N}^d$*, define the differential operator* D^k *by*

$$D^k = \frac{\partial^{k_1+\ldots+k_d}}{\partial x_1^{k_1} \ldots \partial x_d^{k_d}}.$$

Let $\mathcal{F} = \mathcal{F}(\beta, c_1, c_2)$ *be the class of real valued, continuous functions* f *on* $\mathcal{X}$ *with uniformly bounded partial derivatives of order* $k \leq \lfloor\beta\rfloor$ *(the greatest integer smaller than* β*),*

$$\max_{k_1+\ldots+k_n \leq \lfloor\beta\rfloor} \max_{x \in \mathcal{X}} \left|D^k f(x)\right| \leq c_1,$$

and which highest partial derivatives are Lipschitz of order $\beta - \lfloor\beta\rfloor$*,*

$$\max_{k_1+\ldots+k_n=\lfloor\beta\rfloor} \max_{x,y \in \mathcal{X},\ x \neq y} \frac{|D^k f(x) - D^k f(y)|}{\|x-y\|^{\beta-\lfloor\beta\rfloor}} \leq c_2.$$

The constants c_1 *and* c_2 *are independent of* f*. Such classes have covering numbers (Kolmogorov and Tichomirov, 1961; van der Vaart and Wellner, 1996)*

$$\log N(\varepsilon, L_\infty, \mathcal{F}) \leq C_d \left(\frac{1}{\varepsilon}\right)^{d/\beta},$$

for every $\varepsilon > 0$ *and some constant* C_d *depending on the dimension* d *and the constants* c_1 *and* c_2*, but not on* ε*. Applying the theorem with* $p = d/\beta$*, we obtain rates between* $n^{-\beta/(2\beta+d)}$ *(for* $\alpha = 0$*) and* $n^{-\beta/(d+\beta)}$ *(for* $\alpha = +\infty$*).*

Another example is the case where $\mathcal{F}$ *is a subset of a RKHS. For instance, let* $\mathcal{H}$ *be the RKHS corresponding to the Gaussian kernel* $K(x,y) = \exp(-\|x-y\|^2/\sigma^2)$ *and let* $\|f\|$ *be the norm of* f *in* $\mathcal{H}$*. For* $\mathcal{F} = \mathcal{F}_R = \{f \in \mathcal{H} : \|f\| \leq R\}$*, Zhou (2003) proves that, for* $\mathcal{X} = [0,1]^d$*, fixed* R *and fixed scale parameter* σ*, the entropy bound*

$$\log N(\varepsilon, L_\infty, \mathcal{F}) \leq C_d \log^{d+1}\left(\frac{R}{\varepsilon}\right)$$

for some $C_d < \infty$ *and the rates of convergence range between* $\sqrt{\log^{d+1}(n)/n}$ *(*$\alpha = 0$*) and* $\log^{d+1}(n)/n$ *(*$\alpha = \infty$*). See also the results of Guo et al. (2002).*

Acknowledgments

The authors gratefully acknowledge the support of NSF, the first author through grant DMS-0434383 and the second author through grant DMS-0706829.

Appendix A. Proof of Proposition 4

First we compute

$$\begin{aligned}
\inf_{z\le -1} r_{\eta,\phi_d}(z) &= \frac{\eta}{d}, \\
\inf_{-1\le z\le -\delta} r_{\eta,\phi_d}(z) &= \frac{\eta}{d}\mathbf{1}[\eta\le d]+\left(\frac{\delta}{d}\eta+1-\delta\right)\mathbf{1}[\eta>d] \\
\inf_{-\delta\le z\le 0} r_{\eta,\phi_d}(z) &= \mathbf{1}[\eta\ge d]+\left(\frac{\delta}{d}\eta+1-\delta\right)\mathbf{1}[\eta<d] \\
\inf_{0\le z\le \delta} r_{\eta,\phi_d}(z) &= \mathbf{1}[\eta\le 1-d]+\left(1+\frac{\delta}{d}-\delta-\frac{\delta}{d}\eta\right)\mathbf{1}[\eta>1-d] \\
\inf_{\delta\le z\le 1} r_{\eta,\phi_d}(z) &= \frac{1-\eta}{d}\mathbf{1}[\eta>1-d]+\left(1+\frac{\delta}{d}-\delta-\frac{\delta}{d}\eta\right)\mathbf{1}[\eta\le 1-d] \\
\inf_{z\ge 1} r_{\eta,\phi_d}(z) &= \frac{1-\eta}{d}
\end{aligned}$$

It is now easy to verify that

$$\begin{aligned}
H_{-1}(\eta) &= \inf_{z<-\delta} \eta\phi_d(z)+(1-\eta)\phi_d(-z) \\
&= \frac{\eta}{d}\mathbf{1}[\eta<d]+\left(\frac{\delta}{d}\eta+1-\delta\right)\mathbf{1}[\eta\ge d]
\end{aligned}$$

so that

$$\begin{aligned}
&H_{-1}(\eta)-H(\eta)= \\
&\left(\frac{\delta}{d}\eta-\delta\right)\mathbf{1}[d\le\eta\le 1-d]+\left(\frac{1+\delta}{d}\eta+1-\delta-\frac{1}{d}\right)\mathbf{1}[\eta>1-d]
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\xi_{-1}(\eta) &= \eta-\xi(\eta) \\
&= (\eta-d)\mathbf{1}[d\le\eta\le 1-d]+(2\eta-1)\mathbf{1}[\eta>1-d]
\end{aligned}$$

and we see that

$$\frac{\delta}{d}\xi_{-1}(\eta)\le H_{-1}(\eta)-H(\eta)$$

for all $0<\delta\le 1$. Next, we compute

$$\begin{aligned}
H_{\mathbb{R}}(\eta) &= \inf_{|z|\le\delta} \eta\phi_d(z)+(1-\eta)\phi_d(-z) \\
&= \left(1-\delta+\frac{\delta}{d}\eta\right)\mathbf{1}[\eta<d]+\mathbf{1}[d\le\eta\le 1-d] \\
&\qquad +\left(1-\delta+\frac{\delta}{d}-\frac{\delta}{d}\eta\right)\mathbf{1}[\eta>1-d]
\end{aligned}$$

and

$$H_{®}(\eta) - H(\eta) = \left(1 - \delta - \frac{1-\delta}{d}\eta\right)\mathbf{1}[\eta < d] + \left(1 - \delta - \frac{1-\delta}{d} + \frac{1-\delta}{d}\eta\right)\mathbf{1}[\eta > 1-d].$$

Since

$$\begin{aligned}\xi_{®}(\eta) &= d - \xi(\eta) \\ &= (d-\eta)\mathbf{1}[\eta < d] + (d-1+\eta)\mathbf{1}[\eta > 1-d]\end{aligned}$$

we find that

$$\frac{\delta}{d}\xi_{®}(\eta) \le H_{®}(\eta) - H(\eta)$$

provided $0 < \delta \le 1/2$. Finally, we find that

$$\begin{aligned}H_1(\eta) &= \inf_{z>\delta} \eta\phi_d(z) + (1-\eta)\phi_d(-z) \\ &= \frac{1-\eta}{d}\mathbf{1}[\eta > 1-d] + \left(\frac{\delta}{d} + 1 - \delta - \frac{\delta}{d}\eta\right)\mathbf{1}[\eta \le 1-d]\end{aligned}$$

and consequently

$$H_1(\eta) - H(\eta) = \left(1 - \delta + \frac{\delta}{d} - \frac{\delta}{d}\eta - \frac{\eta}{d}\right)\mathbf{1}[\eta < d] + \left(\frac{\delta}{d} - \delta - \frac{\delta}{d}\eta\right)\mathbf{1}[d \le \eta \le 1-d].$$

Now,

$$\begin{aligned}\xi_1(\eta) &= 1 - \eta - \xi(\eta) \\ &= (1-2\eta)\mathbf{1}[\eta < d] + (1-\eta-d)\mathbf{1}[d \le \eta \le 1-d],\end{aligned}$$

and we find that

$$\frac{\delta}{d}\xi_1(\eta) \le H_1(\eta) - H(\eta)$$

provided $0 < \delta \le 1$.

We now verify the second claim of Proposition 4. Assume that $d \le \delta \le 1-d$.

First we consider the case $\eta < d$. Then

$\xi_{-1}(\eta) \le H_{-1}(\eta) - H(\eta)$ holds trivially.

$\xi_{®}(\eta) \le H_{®}(\eta) - H(\eta) \iff (1-\delta-d)\eta \le (1-\delta-d)d$. As $\eta \le d$, we need that $\delta \le 1-d$.

$\xi_1(\eta) \le H_1(\eta) - H(\eta) \iff (1+\delta-2d)\eta \le \delta(1-d)$. As $\eta \le d$, we need that $(1+\delta-2d)d \le \delta(1-d)$, equivalently, $(\delta-d)(1-2d) \ge 0$.

Next, if $d \le \eta \le 1-d$, we see that

$\xi_{-1}(\eta) \leq H_{-1}(\eta) - H(\eta) \iff (\delta - d)\eta \geq d(\delta - d)$.

$\xi_{\circledR}(\eta) \leq H_{\circledR}(\eta) - H(\eta)$ holds trivially.

$\xi_1(\eta) \leq H_1(\eta) - H(\eta) \iff (\delta - d)\eta \leq (1-d)(\delta - d)$.

Finally, if $\eta > 1 - d$, we find that

$\xi_{-1}(\eta) \leq H_{-1}(\eta) - H(\eta) \iff (1 + \delta - 2d)\eta \geq (1 + d\delta - 2d)$. For $\eta \geq 1 - d$ this holds provided $(1 + \delta - 2d)(1 - d) \geq (1 + d\delta - 2d) \iff (\delta - d)(1 - 2d) \geq 0$.

$\xi_{\circledR}(\eta) \leq H_{\circledR}(\eta) - H(\eta) \iff (1 - \delta - d)\eta \geq (1 - d)(1 - \delta - d)$.

$\xi_1(\eta) \leq H_1(\eta) - H(\eta)$ holds trivially.

This concludes the proof of the second claim, since $d \leq \delta \leq 1 - d$. The last claim for the case $(\delta, d) = (0, 1/2)$ follows as well from the preceding calculations.

References

J. Y. Audibert and A. B. Tsybakov. Fast learning rates for plug-in classifiers under margin conditions. *Annals of Statistics*, 35(2):608–633, 2007.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

G. Blanchard, O. Bousquet, and P. Massart. Statistical performance of support vector machines. *Annals of Statistics*, 36(2):489–531, 2008.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: a survey of recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

S. Boucheron, O. Bousquet, and G. Lugosi. Introduction to statistical learning theory. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures in Machine Learning*, pages 169–207. Springer, 2006.

A. Bounsiar, E. Grall, and P. Beauseroy. A kernel based rejection method for supervised classification. *International Journal of Computational Intelligence*, 3(4):312–321, 2006.

C.K. Chow. On optimum error and reject trade-off. *IEEE Transactions on Information Theory*, 16: 41–46, 1970.

D. Cox and F. O'Sullivan. Asymptotic analysis of penalized likelihood and related estimators. *Annals of Statistics*, 18:1676–1695, 1990.

G. Fumera and F. Roli. Suppport vector machines with embedded reject option. In S. Lee and A. Verri, editors, *Pattern Recognition with Support Vector Machines*, volume 2388, pages 68–82. Springer, 2002.

G. Fumera and F. Roli. Analysis of error-reject trade-off in linearly combined multiple classifiers. *Pattern Recognition*, 37:1245–1265, 2004.

G. Fumera, F. Roli, and G. Giacinto. Reject option with multiple thresholds. *Pattern Recognition*, 33:2099–2101, 2000.

G. Fumera, I. Pillai, and F. Roli. Classification with reject option in text categorisation systems. In *Proceedings of the 12th International Conference on Image Analysis and Processing*, pages 582–587. IEEE Computer Society, 2003.

M. Golfarelli, D. Maio, and D. Maltoni. On the error-reject trade-off in biometric verification systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:786 –796, 1997.

Y. Guo, P.L. Bartlett, J. Shawe-Taylor, and R.C Williamson. Covering numbers for support vector machines. *IEEE Transactions on Information Theory*, 48(1):239 – 250, 2002.

L. Györfi, Z. Györfi, and I. Vajda. Bayesian decision with rejection. *Problems of Control and Information Theory*, 8:445–452, 1978.

L. K. Hansen, C. Lissberg, and P. Salamon. The error-reject tradeoff. *Open Systems and Information Dynamics*, 4:159–184, 1997.

R. Herbei and M. H. Wegkamp. Classification with reject option. *Canadian Journal of Statistics*, 4 (4):709–721, 2006.

G.S̃. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

A.N. Kolmogorov and V.M. Tichomirov. ε-entropy and ε-capacity of sets in functional spaces. *American Mathematical Society Translations*, 17:277–364, 1961.

C.W. Landgrebe, D.M.J. Tax, P. Paclik, and R.P.W. Duin. The interaction between classification and reject performance for distance-based reject-option classifiers. *Pattern Recognition Letters*, 27(8):908–917, 2006.

P. Massart. *Concentration Inequalities and Model Selection*, volume 1896. Springer, 2007.

B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

I. Steinwart and C. Scovel. Fast rates for support vector machines using gaussian kernels. *Annals of Statistics*, 35(2):575–607, 2007.

B. Tarigan and S. A. van de Geer. Classifiers of support vector machine type with ℓ_1 complexity regularization. *Bernoulli*, 12(6):1045–1076, 2006.

F. Tortorella. Reducing the classification cost of support vector classifiers through an ROC-based rejection rule. *Pattern Analysis and Applications*, 7:128 – 143, 2004.

A. B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32: 135–166, 2004.

A.W. van der Vaart and J.A. Wellner. *Weak Convergence and Empirical Processes*. Springer, 1996.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32:56–85, 2004.

D.X. Zhou. Capacity of reproducing kernel spaces in learning theory. *IEEE Transactions on Information Theory*, 49(7):1743–1752, 2003.

Journal of Machine Learning Research 9 (2008) 1841-1870 Submitted 11/07; Revised 6/08; Published 8/08

Learning Balls of Strings from Edit Corrections*

Leonor Becerra-Bonache LEONOR.BECERRA-BONACHE@YALE.EDU
Department of Computer Science
Yale University
51 Prospect Street
New Haven, CT, 06511, USA

Colin de la Higuera CDLH@UNIV-ST-ETIENNE.FR
Jean-Christophe Janodet JANODET@UNIV-ST-ETIENNE.FR
Frédéric Tantini FREDERIC.TANTINI@UNIV-ST-ETIENNE.FR
Universities of Lyon
Laboratoire Hubert Curien
18 rue du Professeur Benoît Lauras
42000 Saint-Étienne, France

Editor: Rocco Servedio

Abstract

When facing the question of learning languages in realistic settings, one has to tackle several problems that do not admit simple solutions. On the one hand, languages are usually defined by complex grammatical mechanisms for which the learning results are predominantly negative, as the few algorithms are not really able to cope with noise. On the other hand, the learning settings themselves rely either on too simple information (text) or on unattainable one (query systems that do not exist in practice, nor can be simulated). We consider simple but sound classes of languages defined via the widely used edit distance: the balls of strings. We propose to learn them with the help of a new sort of queries, called the correction queries: when a string is submitted to the Oracle, either she accepts it if it belongs to the target language, or she proposes a correction, that is, a string of the language close to the query with respect to the edit distance. We show that even if the good balls are not learnable in Angluin's MAT model, they can be learned from a polynomial number of correction queries. Moreover, experimental evidence simulating a human Expert shows that this algorithm is resistant to approximate answers.

Keywords: grammatical inference, oracle learning, correction queries, edit distance, balls of strings

1. Introduction

Do you know how many *Nabcodonosaur* were kings of Babylon? And do you know when *Arnold Shwartzeneger* was born? Just two decades ago, you would have had to consult encyclopedias and Who's Who dictionaries in order to get answers to such questions. At that time, you may have needed this information in order to participate to quizzes and competitions organized by famous magazines during the summers, but because of *these* questions, you might possibly have missed the very first prize. Why?... Nowadays, everything has changed: you naturally use the Web, launch

*. This paper is an extended version of "Learning Balls of Strings with Correction Queries" presented at the 2007 European Conference in Machine Learning (ECML'07).

your favorite search engine, type two keywords, follow three links and note down the answers. In this particular case, you discover...that no king of Babylon was called *Nabcodonosaur* but two *Nab***uc**h*odonos***o***r*'s reigned there many centuries ago. Again, the day *Arnold Shwartzeneger* was born is not clear, but it is easy to check that *Arnold S***c***hwa***rz***ene***g***ger* was born in 1947, July 30^{th}.

So you would probably win today the great competitions of the past. Indeed, the actual search engines are able to propose *corrections* when a keyword is not frequent. Those corrections are most often reliable because the reference dictionary is built from the billions of web pages indexed all over the world. Hence, a search engine is playing the role of an imperfect but powerful Oracle, able to validate a relevant query by returning relevant documents, but also to correct any suspect query. Such an Oracle is able to answer to what we shall call *correction queries*.

The first goal of this paper is to show, from a theoretical standpoint, that the concept of correction query allows one to get new challenging results in the field of Active Learning. In this framework, developed by Angluin in the 80's (Angluin, 1987b), a Learner (He) has access to an Oracle (She) that knows a concept he must discover. To this purpose, he submits different kinds of queries (e.g., Correction Queries) and she has to answer without lying. The game ends when he guesses the concept. Query-based Learners are often interesting from a practical viewpoint. For instance, instead of requiring a human expert to label huge quantities of data, this expert could be asked by the Learner, in an interactive situation, to provide a small amount of targeted information.

The second goal of this paper is to provide evidence that correction queries are suitable for this kind of real-life applications. However, assuming that the Oracle is a human expert introduces new constraints. On one hand, it is inconceivable to ask *a polynomial* number of queries: this may still be too much for a human. So the learning algorithm should aim at minimizing the number of queries even if we must pay for it with a worse time complexity. On the other hand, a human being (or even the Web) is fallible. Therefore the learning algorithm should also aim at learning functions or languages from *approximate* corrections.

In the above Web example, the search engine uses the frequency of words to propose corrections. In consequence, correct words (e.g., *malophile* = someone who loves apples) are sometimes subject to a correction (e.g., *halophile* = a cell which thrives in environments with high concentrations of salt). Another key point is the distance used to find a closest correct string; it is a variant of the *edit distance*, also called the *Levenshtein distance*, which measures the minimum number of deletion, insertion or substitution operations needed to transform one string into another (Levenshtein, 1965; Wagner and Fisher, 1974). This distance have been used in many fields including Computational Biology (Gusfield, 1997; Durbin et al., 1998), Language Modelling (Amengual et al., 2001; Amengual and Dupont, 2000) and Pattern Recognition (Navarro, 2001; Chávez et al., 2001).

Edit distance appears in specific Grammatical Inference problems, in particular when one wants to learn languages from noisy data (Tantini et al., 2006). In this domain, the classes of languages studied are not defined following the Chomsky Hierarchy. Indeed, even the easiest level of this hierarchy, the class of regular languages, is not at all robust to noise, since it includes all the parity functions, which can be defined as regular languages and are not learnable in the presence of noise (Kearns and Li, 1993). In order to avoid this difficulty, we shall consider only special finite languages, that may seem elementary to formal language theoreticians, but are relevant for topologists and complex for combinatorialists: the *balls of strings*.

Balls of strings are formed by choosing one specific string, called the centre, and all its neighbours up to a given length for the edit distance, called the radius. From a practical standpoint, balls of strings appear in a variety of settings: in approximate string matching tasks, the goal is to find all

close matches to some target string (Navarro, 2001; Chávez et al., 2001); in noisy settings, garbled versions of an unidentified string are given and the task is to recover the original string (Kohonen, 1985); when using dictionaries, the task can be described as that of finding the intersection between two languages, the dictionary itself and a ball around the target string (Schulz and Mihov, 2002); in the field of bioinformatics, extracting valid models from large data sets of DNA or proteins can involve looking for substrings at distance less than some given bound, and the set of these approximate substrings can also be represented by balls (Sagot and Wakabayashi, 2003).

Hence, in this paper, we study the problem of identifying balls of strings from correction queries. In Section 2, we present the motivations of our work; we discuss why noise is a problem (2.1), which queries should be used to learn languages (2.2), and the relevance of a fallible Oracle in real applications (2.3). Definitions are given in Section 3, where we pay special attention to the definitions of edit distance (3.1), balls of strings (3.2), and correction queries (3.3). On one hand, we prove that the balls are not learnable with Angluin's membership and equivalence queries, and on the other hand, that the deterministic finite automata are not learnable with correction queries.

The main result of the paper is shown in Section 4. It consists of a polynomial time algorithm that infers any ball from correction queries. We explain some technical results (4.1), and we present the algorithm (4.2). An important question is raised concerning the fact that only good balls can be learned with a polynomial number of correction queries (4.3). In Section 5, we study the effectiveness of our algorithm in more practical situations. First, we are concerned with the case where the Oracle is fallible (5.1). Next, we try to minimize the number of queries asked, considering the fact that the expensive resource is the expert playing the part of the Oracle, not the machine making the computations (5.2). We conclude in Section 6.

2. Motivations and Related Work

Several questions need to be addressed before tackling the core of the problem.

2.1 Why is it Hard to Learn Languages in Noisy Settings?

Languages can either be generated, recognized or defined by mechanisms like regular expressions, finite state automata or formal grammars (Harrison, 1978; Sakarovich, 2004; Salomaa, 2006). Alternatively equations can define properties that the strings in the language should verify (Clark et al., 2006). The techniques enabling to learn such formalisms are known as grammatical inference, and new algorithms are developed all the time. But there is one issue that is systematically problematic for such algorithms: that of dealing with noise.

Results obtained in the field of grammatical inference show that learning in noisy situations is hard (de la Higuera, 2006). Some attempts to deal with this problem can be found, for example, in the GOWACHIN (Lang et al., 1998) and GECCO competitions (Lucas, 2004), where the problem of learning DFA from noisy examples was the main challenge.

Noise over strings can, in the simplest case, just affect the labels: a string in the language will be classified as not belonging, whereas a string can be labeled inside the language when it is not. It is known since (Trakhtenbrot and Barzdin, 1973; Angluin, 1978) that with even small doses of noise, learning automata is hard.

The second sort of noise that one may encounter with strings, which is possibly most characteristic here, consists in having the strings slightly modified through some noisy channel. This type of

noise is invariably described by the edit distance (Levenshtein, 1965): individual symbols appear or disappear, or even are transformed into different ones.

Again, for the edit noise, the typical classes of languages belonging to the Chomsky hierarchy (Chomsky, 1957; Sakarovich, 2004) are far from robust. Consider for instance the set of strings over the alphabet $\{0,1\}$ whose parity of 0's is identical to the parity of 1's (see Figure 1). This typical

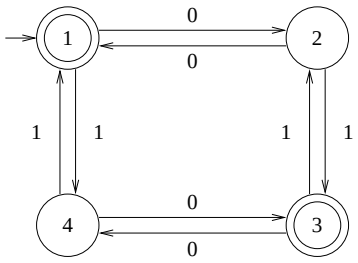


Figure 1: An automaton recognizing $\{w \in (0+1)^* : |w|_0 \bmod 2 = |w|_1 \bmod 2\}$.

regular language is clearly very sensitive to the noise: if any symbol is inserted or deleted in a string, the string will cease to belong to the language; and conversely, any string out of the language will be transformed into a string from the language, as the parity of either of the letters will change.

Unfortunately, the picture is even less clear with other regular languages such as 0^*1^* and $(010)^*$, or higher languages in the Chomsky hierarchy such as $\{ww : w \in (0+1)^*\}$ or the set of palindromes or $\{0^n1^n2^n : n \geq 0\}$. Indeed, these languages are sparse in the set of all strings, so trying to learn them from noisy data is like looking for a needle in a haystack: no string seems to belong to the target anymore.

The reader may think that probably, all these textbook languages are not relevant in practice. In which case, studying their learnability in the presence of noise would not be significative. Nevertheless, concerning randomly drawn regular languages, the picture is not better: the website developed by Coste et al. (1998) shows that despite a decade of efforts, no convincing solution has been yet found to take into account the noise during the learning process.

Therefore, if we are to learn languages in a noisy setting where the noise is modelled through the edit distance, we think that it is necessary to consider other classes of languages that could be much better adapted to this type of noise. The balls of strings are an example of such languages.

2.2 What Queries Should we Use?

Learning with queries was introduced by Angluin in order to provide a firm mathematical background to machine learning in a non statistical setting (Angluin, 1987b). In this paradigm, both positive and negative results are relevant. Indeed, if one cannot learn using a polynomial number of questions, then one cannot do it from data that one gets without choice from the environment. In this setting the questions are called queries and they are asked to a perfect abstract machine, called Oracle.

Several types of queries have been studied, and some classes were proved to be learnable from specific combination of queries (see Angluin, 2004, for a survey). The best known and most important of such positive results is that deterministic finite state automata are polynomially learnable

from a combination of membership queries and strong equivalence queries (Angluin, 1987a). The corresponding definitions will be given in Section 3.3.

We argue that equivalence queries are not realistic for the intended applications, and we choose to use the recently introduced correction queries instead (Becerra-Bonache and Yokomori, 2004). When making a correction query, we submit a string to the Oracle who answers YES if the string is in the target language, and if it is not then the Oracle returns a string from the language that is closest to the query. This string is called the correction.

In order to give an introductory intuition, let us consider the case where we want to learn disks in the plane using the Euclidean distance. Instead of learning from examples (with the possibility of them being labeled), let us suppose we have access to an Oracle that will answer the following query: a point is proposed, and is returned either the answer YES or a correction of this point, that is, the closest point in the disk.

Then we can proceed in three stages to learn a disk of centre O and radius R with correction queries as shown in Figure 2:

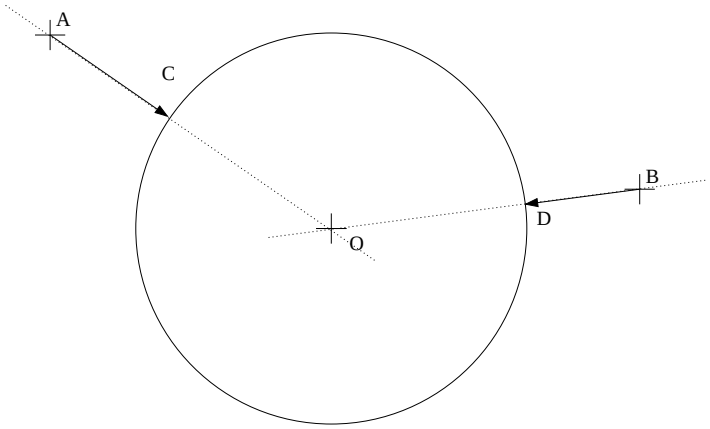


Figure 2: Three stages are sufficient to learn the disks of $\mathbb{R}^2$ with correction queries: (1) find two points A and B outside of the disk haphazardly using correction queries; (2) ask the Oracle for the corrections of A and B, which will result in C and D, respectively; (3) use a ruler to deduce the centre O and a compass to draw the circle.

1. We start by finding two points A and B outside of the disk we want to identify. Looking for them haphazardly by asking to the Oracle if such or such a point is in the disk is enough: intuitively, we are going to find them with very few queries.

2. We ask the Oracle for the corrections of A and B. Concerning A, the Oracle is going to return a point C inside the disk, as close as possible to A. Clearly, this point is at the intersection of the segment $[OA]$ and the boundary circle of the target disk. Likewise, let D be the correction of B.

3. We draw the lines (AC) and (BD) with a ruler: they intersect in O. Then we can draw the circle with a compass. We get the radius by measuring the distance between O and C.

Hence, it is easy to learn the balls of $\mathbb{R}^2$ with very few correction queries. Now, focusing on balls of strings, we may hope that the previous approach is good and try to reproduce it.

However, building the centre of a ball from strings on its periphery is difficult for at least two reasons. On one hand, (Σ^*, d) is a metric space with no vector space as underlying structure. This is similar to the case where we were trying to learn the disks of the plane with just a compass but no ruler.[1] On the other hand, the problem is formally *hard*:

Theorem 1 (de la Higuera and Casacuberta 2000) *Given a finite set of strings $W = \{w_1, \ldots, w_n\}$ and a constant K, deciding whether a string $u \in \Sigma^*$ exists such that $\max_{w \in W} d(u,w) < K$ (respectively $\sum_{w \in W} d(u,w) < K$) is $\mathcal{NP}$-complete.*

Therefore, we will have to study the balls in more detail and make the best possible use of the correction queries, so as not to build the centres from scratch.

2.3 Why might the Oracle be Fallible?

Above we argued that the active learning model was based on the strong assumption of a perfect Oracle. This corresponded to a reasonable assumption when dealing with mathematics and with the objective of being in a favorable setting in which negative results could be blamed on the complexity of the task and not on the adversarial nature of the Oracle.

But in recent years, the active learning setting (corresponding to learning from an Oracle) has been accepted as a plausible setting for real applications. Indeed we are faced with huge quantities of unlabeled data. Choosing which data is to receive attention by an expert (human or machine) is a difficult question. Interactive learning sessions, where the learning algorithm asks for specific information during runtime, is an interesting alternative to deal with such problems.

A typical example is system SQUIRREL (Carme et al., 2007) which induces a web wrapper through interaction with a human user. Another case is that of testing hardware (Hagerer et al., 2002): the specifications of the software correspond to the Oracle which can then allow to check if the constructed item obeys to the specifications. In both examples the Oracle is fallible: in the second one because testing equivalence is done through sampling.

A third situation in which active learning can be useful corresponds to that of rendering intelligible some black box learned through some statistical machine learning method. Indeed, even if hyper-planes (Clark et al., 2006) or recurrent neural networks (Giles et al., 2001) are difficult to interpret, one can try to use the learned numerical models as Oracles in an active learning algorithm whose result might be some rule based classifier (de la Higuera, 2006).

3. Definitions

An *alphabet* Σ is a finite nonempty set of symbols called *letters*. For the sake of clarity, we shall use $0, 1, 2, \ldots$ as letters in our examples and write $a, b, c, \ldots$ to denote variables for letters in an alphabet. A *string* $w = a_1 \ldots a_n$ is any finite sequence of letters. We write Σ^* for the set of all strings over Σ and λ for the empty string. Let $a \in \Sigma$, $|w|$ be the length of w and $|w|_a$ the number of occurrences of

1. Actually, this is still possible: a theorem due to Mohr (1672), rediscovered by Mascheroni (1797), states that every construction with a ruler and a compass can also be done with a compass only. We know an algorithm that uses 14 circles to learn a disk of the plane. If the reader knows a better method, please contact us!

a in w. We say that a string u is a *subsequence* of v, denoted $u \preceq v$, if $u = a_1 \dots a_n$ and there exist $u_0, \dots, u_n \in \Sigma^*$ such that $v = u_0 a_1 u_1 \dots a_n u_n$. A *language* is any subset $L \subseteq \Sigma^*$. Let $\mathbb{N}$ be the set of non negative integers. For all $k \in \mathbb{N}$, let $\Sigma^k = \{w \in \Sigma^* : |w| = k\}$ and $\Sigma^{\leq k} = \{w \in \Sigma^* : |w| \leq k\}$. Let $\mathbb{R}$ denote the set of real numbers. We say that a real number $\rho \in \mathbb{R}$ is *irrational* if $|\rho| \neq \frac{p}{q}$ for all $p, q \in \mathbb{N}$.

3.1 Edit Distance

The *edit distance* $d(w, w')$ between two strings w and w' is the minimum number of *edit operations* needed to transform w into w' (Levenshtein, 1965).

More precisely, we say that *w rewrites to w' in one step*, written $w \to w'$, if either

1. $w = uav$ and $w' = uv$ (*deletion* of a letter), or
2. $w = uv$ and $w' = uav$ (*insertion* of a letter), or
3. $w = uav$ and $w' = ubv$ (*substitution* of a letter by another letter),

where $u, v \in \Sigma^*$, $a, b \in \Sigma$ and $a \neq b$.

Let $\xrightarrow{k}$ denote a *rewriting derivation* made of k rewriting steps. The edit distance $d(w, w')$ is the minimum $k \in \mathbb{N}$ such that $w \xrightarrow{k} w'$. For instance, $d(0100, 001) = 2$ since $0\underline{1}00 \to 00\underline{0} \to 001$ and rewriting 0100 into 001 cannot be achieved with less than two steps. Notice that $d(w, w')$ can be computed in time $O(|w| \cdot |w'|)$ by means of dynamic programming (Wagner and Fisher, 1974).

The following basic property states that $d(w, w')$ is at least the number of insertions needed to equalize the lengths of w and w':

Proposition 2 *For all $w, w' \in \Sigma^*$, $d(w, w') \geq \big||w| - |w'|\big|$. Moreover, $d(w, w') = \big||w| - |w'|\big|$ iff ($w \preceq w'$ or $w' \preceq w$).*

In all the parts of this paper but in Section 5.2.1, we shall use the standard edit distance defined above. However, for practical reasons, people often use variants of this definition. Sometimes, new edit operations are defined such as the exchange of two adjoining letters in a string. And often, the edit operations are weighted. We shall give more details when needed.

3.2 Balls of Strings

It is well-known that the edit distance is a *metric* (Crochemore et al., 2007), so it conveys to Σ^* the structure of a *metric space*.

Definition 3 (Ball of Strings) *The ball of centre $o \in \Sigma^*$ and radius $r \in \mathbb{N}$, denoted $B_r(o)$, is the set of all the strings whose distance to o is at most r:*

$$B_r(o) = \{w \in \Sigma^* : d(o, w) \leq r\}.$$

For instance, if $\Sigma = \{0, 1\}$, then $B_1(10) = \{0, 1, 00, 10, 11, 010, 100, 101, 110\}$ and $B_r(\lambda) = \Sigma^{\leq r}$ for all $r \in \mathbb{N}$.

The previous example illustrates the fact that the number of strings in a ball grows exponentially with the radius. Experimentally (see Table 1), we clearly notice that for center strings of fixed length, the average number of strings is more than twice larger when the radius is incremented by 1. This

Length of the centre	Radius					
	1	2	3	4	5	6
0	3.0	7.0	15.0	31.0	63.0	127.0
1	6.0	14.0	30.0	62.0	126.0	254.0
2	8.6	25.6	56.5	119.7	246.8	501.6
3	10.8	41.4	101.8	222.8	468.6	973.0
4	13.1	61.4	173.8	402.9	870.9	1850.8
5	16.3	91.0	285.1	698.5	1584.4	3440.9
6	17.9	125.8	441.2	1177.5	2771.3	6252.9
7	21.2	166.9	678.0	1908.8	4835.8	11233.5
8	24.3	200.2	1034.2	3209.9	8358.1	19653.6
9	26.0	265.4	1390.9	5039.6	13677.8	34013.1

Table 1: Average number of strings in a ball. The alphabet has 2 letters. Each value is computed over 20 random centres (possibly the same).

combinatorial explosion occurs as soon as $|\Sigma| \geq 2$, although we leave open the question of finding a general formula that would assess the volume of any ball $B_r(o)$.

The combinatorial explosion noted before raises the problem of the representation scheme that we should use to learn the balls, that is to say, the format of the output space of any learning algorithm. Basically, we need representations whose size is "reasonable", which is not the case of an exhaustive enumeration. An alternative representation could be based on automata, since the balls of strings are finite and thus regular languages.

It is not difficult to see that every ball $B_r(o)$ is recognized by a *non deterministic finite automaton with λ-transitions* having $O(|o| \cdot r)$ states. However, the non deterministic automata are bad candidates from the learning standpoint. Indeed, they are not learnable in most paradigms (Angluin and Kharitonov, 1995; de la Higuera, 1997).

The corresponding *deterministic finite automata* (DFA) do not have this drawback. However, experiments show that these DFA often have an exponential number of states. More precisely, several efficient algorithms exist to build a DFA that recognizes $B_r(o)$ (Ukkonen, 1985; Melichar, 1995; Schulz and Mihov, 2002). For instance, Schulz and Mihov (2002) have recently introduced the so-called *Levenshtein automaton*. Denoting by $n(o,r)$ the number of states of this automaton, they state: $n(o,1) = O(5 \cdot |o|)$, $n(o,2) = O(30 \cdot |o|)$, $n(o,3) = O(180 \cdot |o|)$, $n(o,4) = O(1353 \cdot |o|)$. Basically, $n(o,r)$ is linear in $|o|$ but exponential in r (In their construction, the size of the alphabet only plays a role in the number of transitions, not in the number of states).

Unfortunately, proving that the minimal DFA has the same property is a challenging combinatorial problem. So we only claim here:

Conjecture 4 *The minimal* DFA *recognizing the ball $B_r(o)$ has $\Omega(2^r \cdot |o|)$ states in the worst case.*

On the other hand, why not represent the ball $B_r(o)$ by the pair (o,r) itself? Indeed, its size is $|o| + \log r$, which is reasonable (Garey and Johnson, 1979). Besides, deciding whether $w \in B_r(o)$ or not is immediate: one only has to (1) compute $d(o,w)$ and (2) check whether this distance is $\leq r$,

which is achievable in time $O(|o| \cdot |w| + \log r)$. Finally, when the alphabet has at least two letters, (o, r) is a unique thus *canonical* representation of $B_r(o)$:

Theorem 5 *If $|\Sigma| \geq 2$ and $B_{r_1}(o_1) = B_{r_2}(o_2)$, then $o_1 = o_2$ and $r_1 = r_2$.*

Proof

- *Claim 1: if $B_{r_1}(o_1) = B_{r_2}(o_2)$, then $|o_1| + r_1 = |o_2| + r_2$.* Indeed, let $w \in \Sigma^{r_1}$, then $d(o_1, o_1 w) = |w| = r_1$ by Proposition 2. So $o_1 w \in B_{r_1}(o_1)$, thus $o_1 w \in B_{r_2}(o_2)$, that is to say, $d(o_1 w, o_2) \leq r_2$. Now $d(o_1 w, o_2) \geq |o_1 w| - |o_2|$ from Proposition 2. So we deduce that $r_2 \geq |o_1 w| - |o_2| = |o_1| + r_1 - |o_2|$. The same reasoning yields $|o_1| + r_1 \geq |o_2| + r_2$.
- *Claim 2: if $|\Sigma| \geq 2$ and $o_2 \not\preceq o_1$, then there exists $w \in \Sigma^*$ such that (1) $|w| = r_1 + |o_1|$ and (2) $o_1 \preceq w$ and (3) $o_2 \not\preceq w$.* Indeed, assume that $\Sigma = \{0, 1, \ldots\}$ and o_2 begins with an 0. Then we define $w = 1^{r_1} o_1$ and get the result.

Theorem itself: Assume that $o_1 \neq o_2$. Then either $o_1 \not\preceq o_2$, or $o_2 \not\preceq o_1$. Suppose that $o_2 \not\preceq o_1$, without loss of generality. By Claim 2, there exists a string w such that (1) $|w| = r_1 + |o_1|$ and (2) $o_1 \preceq w$ and (3) $o_2 \not\preceq w$. As $o_1 \preceq w$, Proposition 2 yields $d(o_1, w) = |w| - |o_1| = r_1$. So $w \in B_{r_1}(o_1)$. On the other hand, $o_2 \not\preceq w$, so Proposition 2 yields $d(o_2, w) > \big||w| - |o_2|\big| = \big|r_1 + |o_1| - |o_2|\big| = r_2$, so $w \notin B_{r_2}(o_2)$. In consequence, $B_{r_1}(o_1) \neq B_{r_2}(o_2)$, that is impossible. Therefore, $o_1 = o_2$, and by Claim 1, $r_1 = r_2$. ■

Notice however that if $\Sigma = \{0\}$, then $B_2(0) = B_3(\lambda) = \{\lambda, 0, 00, 000\}$, for instance.

3.3 Queries

Query learning is a paradigm introduced by Angluin (1987b). Her model brings a Learner and an Oracle into play. The goal of the Learner is to identify the representation of an unknown language, by submitting queries to the Oracle. The latter knows the target language and answers properly to the queries (she does not lie). The Learner is bounded by efficiency constraints at each step of the learning process: the runtime of the Learner to make its next query must be polynomial in the size of the target representation and in the length of the information returned by the Oracle up to that point. Notice that certain types of queries require answers that may be of unbounded length (examples or counter-examples). In that case, it is impossible not to take into account the length of this information in the amount of time and queries the Learner is allowed.

Between the different combinations of queries, one, called MAT (Minimally Adequate Teacher), is sufficient to learn the DFA (Angluin, 1987a). Two kinds of queries are used:

Definition 6 (Membership and Equivalence Queries) *Let Λ be a class of languages on Σ^* and $L \in \Lambda$ a target language known by the Oracle, that the Learner aims at guessing.*

In the case of membership queries, the Learner submits a string $w \in \Sigma^*$ to the Oracle; her answer, denoted by $\mathrm{MQ}(w)$, is either YES if $w \in L$, or NO if $w \notin L$.

In the case of equivalence queries, the Learner submits (the representation of) a language $K \in \Lambda$ to the Oracle; her answer, denoted by $\mathrm{EQ}(K)$, is either YES if $K = L$, or a string belonging to the symmetric difference $\big((K \setminus L) \cup (L \setminus K)\big)$ if $K \neq L$.

Although membership queries and equivalence queries have established themselves as a standard combination, there are real grounds to believe that equivalence queries are too powerful to

exist or even be simulated. From a cognitive point of view, we may imagine that a child could ask to his mother whether some sentence is correct or not (that would be a membership query), but not whether he knows English or not (that would be an equivalence query). As suggested by Angluin (1987a), in practice, we may be able to substitute the equivalence queries with a random draw of strings that are then submitted as membership queries (*sampling*). However, in many cases, sampling is not possible because the relevant distribution is unknown and/or inaccessible (de la Higuera, 2006).

Besides, we will not consider membership queries and equivalence queries together because they do not help to learn balls:

Theorem 7 *Assume* $|\Sigma| \geq 2$. *Let* $m,n \in \mathbb{N}$ *and* $\mathcal{B}_{\leq m,n} = \{B_r(o) : r \leq m, o \in \Sigma^*, |o| \leq n\}$. *Any algorithm that identifies every ball of* $\mathcal{B}_{\leq m,n}$ *with equivalence queries and membership queries necessarily uses* $\Omega(|\Sigma|^n)$ *queries in the worst case.*

Proof Following Angluin (1987b), we describe a malevolent Oracle who forces any method of exact identification using membership and equivalence queries to make $\Omega(|\Sigma|^n)$ queries in the worst case. The Oracle is an Adversary: she changes the target ball during the process of identification in order to penalize the Learner. However, all her answers will have to be consistent with the final ball. Technically, she maintains a set S of all the possible balls. At the beginning, $S = \mathcal{B}_{\leq m,n}$. As long as S contains at least two balls, she proceed as follows: her answer to the equivalence query $L = B_r(o)$ is the counterexample o; her answer to the membership query o is NO; in other words, she always declares that o is not in the target ball. After such an answer, every ball of S that contains o cannot be a possible target anymore, so she eliminates them from S. At this point, many balls might disappear, but only one of centre o and radius 0. As there are $\Omega(|\Sigma|^n)$ such balls in $\mathcal{B}_{\leq m,n}$, the Learner will need $\Omega(|\Sigma|^n)$ queries to identify one of them. ■

It should be noted that if the Learner is given one string from the ball, then he can learn using a polynomial number of membership queries.[2] We shall see that the *correction queries*, introduced below, allow to get round these problems:

Definition 8 (Correction Queries) *Let L be a target language known by the Oracle and w a string submitted by the Learner to the Oracle. Her answer, denoted* $\text{CQ}(w)$*, is either* YES *if* $w \in L$*, or a correction of w with respect to L if* $w \notin L$*, that is a string* $w' \in L$ *at minimum edit distance from w:*

$$\text{CQ}(w) = \textit{one string of } \left\{w' \in L : d(w,w') \textit{ is minimum}\right\}.$$

Notice that other milder definitions of correction queries have been proposed in the literature such as Becerra-Bonache et al. (2006) and Kinber (2008). However, the correction queries defined above can easily be simulated knowing the target language. Moreover, we have seen in the introduction that they naturally exist in real-world applications such as the search engines of the Web. Also, we can note that the correction queries are relevant from a cognitive point of view: there is growing evidence that corrective input for grammatical errors is widely available to children (Becerra-Bonache, 2006).

And last but not least, the correction queries as well as the balls rely on a distance, that foreshadows nice learning results. This is not the case for every class of languages:

2. More precisely, the best algorithm we know uses $O\big(|\Sigma|(|o|+r)\big)$ membership queries.

Theorem 9 *Assume $|\Sigma| \geq 2$. Let $n \geq 2$ and $\mathcal{D}_{\leq n}$ the set of all* DFA *with fewer than n states. Any algorithm that identifies every* DFA *of $\mathcal{D}_{\leq n}$ with correction queries necessarily uses $\Omega(|\Sigma|^n)$ queries in the worst case.*

Proof Remember that the number of states of a DFA is a reasonable measure of its size. Let A_w denote the minimal DFA that recognizes $\Sigma^* \setminus \{w\}$. The reader may check that A_w has $|w|+2$ states (see Figure 3 for an example). So basically, $\{A_w : w \in \Sigma^{n-2}\} \subseteq \mathcal{D}_{\leq n}$. Following Angluin (1987b) again, we describe an Adversary that maintains a set S of all the possible DFA. At the beginning, $S = \mathcal{D}_{\leq n}$. Each time the correction of any string w is demanded, the Adversary answers YES and eliminates $\mathcal{A}_w$ from S (and a lot of other DFA) in order to be consistent. As there are $\Omega(|\Sigma|^n)$ such DFA in $\mathcal{D}_{\leq n}$, identifying one of them requires $\Omega(|\Sigma|^n)$ queries. ■

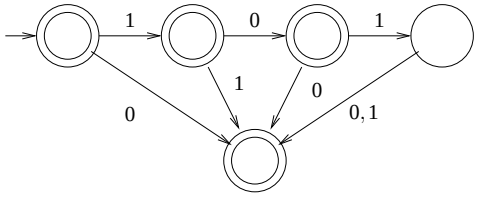


Figure 3: The minimal DFA A_{101} that recognizes $\Sigma^* \setminus \{101\}$ has $5 (= |101|+2)$ states.

4. Identifying Balls of Strings using Corrections

In this section, we propose an algorithm that learns the balls of strings using correction queries. We follow the method described for the disks of the plane. However, several details distinguish the balls of strings and the balls in $\mathbb{R}^2$.

4.1 Technicalities

In this section we introduce four related mathematical results. The first is an analysis of the corrections the Oracle can make. The second corresponds to the definition of the set of the longest strings in a ball (what we call the *upper border* of the ball). The third result is an algorithm allowing to extract the centre of the ball if we are given some elements from this upper border. And finally we explain how to find a string from the upper border using corrections.

4.1.1 A CHARACTERIZATION OF THE CORRECTIONS

When the Learner submits a string outside of a ball to the Oracle, she answers with a string that belongs to the 'circle' delimiting the ball. However, a string often has a lot of different possible corrections, contrarily to what happens in the plane. For instance, the possible corrections for the string 0000 with respect to the ball $B_2(11)$ are $\{00, 001, 010, 100, 0011, 0101, 0110, 1001, 1010, 1100\}$.

By the definition of a correction query, the Oracle will choose one of them arbitrarily, possibly the worst one from the Learner's point of view. Nevertheless, the Oracle's potential malevolence is limited by the following result, that characterizes the set of *all* the possible corrections for a string:

Lemma 10 *Let $B_r(o)$ be a ball and $v \notin B_r(o)$. Then the set of possible corrections of v is exactly $\{u \in \Sigma^* : d(o,u) = r \text{ and } d(u,v) = d(o,v) - r\}$.*

Proof Let $k = d(o,v)$ and consider a derivation from o to v of minimum length: $o \xrightarrow{k} v$. As $v \notin B_r(o)$, we get $k > r$, so this derivation passes through a string w_0 such that $o \xrightarrow{r} w_0 \xrightarrow{k-r} v$. Let us define the set $W = \{w \in \Sigma^* : d(o,w) = r \text{ and } d(w,v) = k-r\}$. Basically, $w_0 \in W$, so $W \neq \emptyset$. Moreover, $W \subseteq B_r(o)$. Now let U denote the set of all the possible corrections of v. We claim that $U = W$. Indeed, let $u \in U$ and $w \in W$. If $d(u,v) > d(w,v)$, then w is a string of $B_r(o)$ that is closer to v than u, so u cannot be a correction of v. On the other hand, if $d(u,v) < d(w,v)$, then as $d(o,v) \leq d(o,u) + d(u,v)$, we deduce that $d(o,u) \geq d(o,v) - d(u,v) > d(o,v) - d(w,v)$. As $d(o,v) = k$ and $d(w,v) = k-r$, we get $d(o,u) > r$, which is impossible since $u \in U \subseteq B_r(o)$. Hence, $d(u,v) = d(w,v) = k-r$. In consequence, all the strings $w \in W$ and corrections $u \in U$ are at the same distance from v, thus $W \subseteq U$. Moreover, we have $d(o,v) \leq d(o,u) + d(u,v)$, so $k \leq d(o,u) + k - r$, thus $d(o,u) \geq r$. As $u \in B_r(o)$, we deduce that $d(o,u) = r$. Finally, as we have stated that $d(u,v) = k-r$, we can conclude that $U \subseteq W$. ■

Here is a geometric interpretation of the result above. Let us define the *segment* $[o,v] = \{w \in \Sigma^* : d(o,w) + d(w,v) = d(o,v)\}$ and the *circle* $C_r(o) = \{w \in \Sigma^* : d(o,w) = r\}$. Lemma 10 states that a string u is a possible correction of v *iff* $u \in [o,v] \cap C_r(o)$. The fact that v has several possible corrections shows that the geometry of Σ^* is very different from that of $\mathbb{R}^2$.

4.1.2 THE BORDERLINE STRINGS OF MAXIMUM LENGTH

We begin by distinguishing the longest strings of any ball:

Definition 11 (Upper Border) *The upper border of a ball $B_r(o)$, denoted $B_r^{max}(o)$, is the set of all the strings that belong to $B_r(o)$ and are of maximum length:*

$$B_r^{max}(o) = \{u \in B_r(o) : \forall w \in B_r(o), |w| \leq |u|\}.$$

For instance, given $\Sigma = \{0,1\}$, we get $B_1^{max}(10) = \{010, 100, 101, 110\}$.

The strings of $B_r^{max}(o)$ are remarkable because they are all built from the centre o by doing r insertions. So from a string $w \in B_r^{max}(o)$, one 'simply' has to guess the inserted letters and delete them to find o again. We get:

Proposition 12 *$w \in B_r^{max}(o)$ iff ($o \preceq w$ and $d(o,w) = |w| - |o| = r$).*

Proof Let us assume that $o \preceq w$ and $d(o,w) = |w| - |o| = r$. Then $w \in B_r(o)$. Let w' be a string such that $|w'| > |w|$. Then, by Proposition 2, $d(o,w') \geq |w'| - |o| > |w| - |o| = r$, so $w' \notin B_r(o)$. Therefore, $w \in B_r^{max}(o)$. Conversely, let $w \in B_r^{max}(o)$. Consider an arbitrary letter $a \in \Sigma$ and the string $a^r o$. Basically, $d(o,a^r o) = r$, so $a^r o \in B_r(o)$. As $w \in B_r^{max}(o)$, we deduce that $|w| \geq |a^r o| = |o| + r$. Therefore, by Proposition 2, $d(o,w) \geq |w| - |o| \geq r$. On the other hand, $r \geq d(o,w)$ holds since $w \in B_r^{max}(o)$. So we deduce that $d(o,w) = |w| - |o| = r$, that also brings $o \preceq w$, by Proposition 2. ■

4.1.3 FINDING THE CENTRE GIVEN STRINGS FROM THE UPPER BORDER

Some strings of $B_r^{max}(o)$ are even more informative. Indeed, let $a \in \Sigma$ be an arbitrary letter. Then $a^r o \in B_r^{max}(o)$. So, if we know r, we can easily deduce o. We claim that the correction queries allow us to get hold of $a^r o$ from any string $w \in B_r^{max}(o)$ by swapping the letters. This is the goal of EXTRACT_CENTRE (see Algorithm 1).

Let us run this procedure on an example. Consider the ball $B_2(11)$. Then it is easy to check that $B_2^{max}(11) = \{0011, 0101, 0110, 0111, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$. Running EXTRACT_CENTRE on the string $w = 0110$ and radius $r = 2$ transforms, at each loop, the i^{th} letter of w to a 0 that is put at the beginning and then submits it to the Oracle. We get:

i	w	w'	CQ(w')	w changes
1	$\underline{0}110$	0110	YES	yes
2	$0\underline{1}10$	0010	0110	no
3	$01\underline{1}0$	0010	0110	no
4	$011\underline{0}$	0011	YES	yes

Therefore, EXTRACT_CENTRE stops with $w = 0011$ and returns $o = 11$ (since $r = 2$).

Algorithm 1 EXTRACT_CENTRE

Require: A string $w = a_1 \dots a_n \in B_r^{max}(o)$, the radius r
Ensure: The centre o of the ball $B_r(o)$

```
x ← a_n        (* x is an arbitrary letter *)
for i = 1 to n do
    Assume w = a_1 ... a_n and let w' = x a_1 ... a_{i-1} a_{i+1} ... a_n
    if CQ(w') = YES then w ← w' end if
end for
Assume w = a_1 ... a_n and return a_{r+1} ... a_n
```

Proposition 13 *Given $w \in B_r^{max}(o)$ and the radius r, Algorithm* EXTRACT_CENTRE *returns o using $O(|o| + r)$ correction queries.*

Proof (Sketch) Let us show that the swapping operation is correct. Consider the string $w = a_1 \dots a_n \in B_r^{max}(o)$ and let $w' = x a_1 \dots a_{i-1} a_{i+1} \dots a_n$ for some $0 \leq i \leq n$. If there exists at least one derivation $o \xrightarrow{r} w$ where the letter a_i of w comes from an insertion in o, then deleting a_i and doing the insertion of a x in front of o yields a string w' that satisfies $o \preceq w'$ and $|w'| = |w|$. By Proposition 2, we get $d(o, w') = |w'| - |o| = |w| - |o| = r$, so by Proposition 12, $w' \in B_r^{max}(o)$ and CQ(w') = YES. On the other hand, if there is no derivation where a_i is introduced by an insertion, then deleting a_i and inserting a x yields a string w' such that $o \not\preceq w'$. By Proposition 2, we get $d(o, w') > |w'| - |o|$. As $|w'| = |w|$, we deduce that $d(o, w') > r$. So $w' \notin B_r^{max}(o)$ and CQ(w') $\neq$ YES. ■

4.1.4 FINDING ONE BORDERLINE STRING OF MAXIMUM LENGTH

Hence, we are now able to deduce the centre of a ball as soon as we know its radius and a string from its upper border. The following technical result is a step towards finding this string (although we have no information about r and $|o|$ yet):

Proposition 14 *Suppose that $\Sigma = \{a_1, \ldots, a_n\}$ and consider the string $v = (a_1 \ \ldots \ a_n)^k$ with $k \geq |o| + r$. Then every correction of v belongs to $B_r^{max}(o)$.*

Proof Let U be the set of all the possible corrections of v. Let us show that $U = B_r^{max}(o)$. As $v = (a_1 \ldots a_n)^k$ with $k \geq |o| + r$, we get $o \preceq v$, so $d(o,v) = |v| - |o|$, by Proposition 2. Let $w \in B_r^{max}(o)$. By Proposition 12, we get $o \preceq w$ and $d(o,w) = |w| - |o| = r$. Moreover, as $v = (a_1 \ldots a_n)^k$ with $k \geq |o| + r$, we get $w \preceq v$. So $d(w,v) = |v| - |w| = |v| - |o| - r = d(o,v) - r$ by Proposition 2. As $d(o,w) = r$ and $d(w,v) = d(o,v) - r$, Lemma 10 yields $B_r^{max}(o) \subseteq U$. Conversely, let $u \in U$. We get $d(o,u) = r$, again by Lemma 10. If $o \preceq u$, then $u \in B_r^{max}(o)$ by Proposition 12. If $o \not\preceq u$, then Proposition 2 yields $d(o,u) > |u| - |o|$, that is to say, $|u| < |o| + r$. But then, $d(u,v) \geq |v| - |u| > |v| - |o| - r = d(w,v)$ for all $w \in B_r^{max}(o)$, so $u \notin U$, that is impossible. Therefore, $U \subseteq B_r^{max}(o)$. ■

If one submits $(a_1 \ldots a_n)^k$ with a sufficiently large k, then one is sure to get a string of $B_r^{max}(o)$. So the last problem is to find such an interesting k. The following lemma states that if one asks to the Oracle the correction of a string made of a lot of 0's, then this correction contains precious informations about the radius and the number of occurrences of 0's in the centre:

Lemma 15 *Consider the ball $B_r(o)$. Let $a \in \Sigma$ be any letter and $j \in \mathbb{N}$ an integer such that $a^j \notin B_r(o)$. Let w denote a correction of a^j. If $|w| < j$, then $|w|_a = |o|_a + r$.*

Proof Let $a^j \notin B_r(o)$ and $w = \mathrm{CQ}(a^j)$. By Lemma 10, we get $d(o,w) = r$ and $d(w,a^j) = d(o,a^j) - r$. As $|w| < |a^j|$, the computation of $d(w,a^j)$ consists in (1) substituting all the letters of w that are not a's, thus doing $|w| - |w|_a$ substitutions, and (2) completing this string with further a's in order to reach a^j, thus doing $j - |w|$ insertions of a's. So we deduce that $d(w,a^j) = |w| - |w|_a + j - |w| = j - |w|_a$. Let us compute $d(o,a^j)$. Clearly, if $|o| \leq |w| < j$, then we can use the same arguments as before and get $d(o,a^j) = j - |o|_a$. Finally, since $d(w,a^j) = d(o,a^j) - r$, we deduce that $j - |w|_a = j - |o|_a - r$, *that is*, $|w|_a = |o|_a + r$.

Now suppose that $|o| > |w|$. Then we cannot use the same arguments as before, because it is possible that $|o| \geq |a^j|$, thus that deletions are needed to compute $d(o,a^j)$. However, this case is impossible. Indeed, consider a derivation $o \xrightarrow{r} w$. Since $|o| > |w|$, there is at least one deletion along this derivation. Now, instead of deleting a letter, suppose that we substitute it by an a and do not change anything else. This leads us to a new derivation $o \xrightarrow{r} w'$ (or $o \xrightarrow{r-1} w'$ if the deleted letter was an a) with $|w'| = |w| + 1$ and $|w'|_a = |w|_a + 1$. Moreover, $d(o,w') \leq r$, thus $w' \in B_r(o)$. Finally, as $|w| < j$, we get $|w'| \leq j$, so with the same arguments as before, only substitutions and insertions are necessary to compute $d(w',a^j)$. More precisely, we have $d(w',a^j) = (|w'| - |w'|_a) + (j - |w'|) = -|w|_a - 1 + j = d(w,a^j) - 1$, thus $d(w',a^j) < d(w,a^j)$. Since $w' \in B_r(o)$, w cannot be a correction of a^j. ■

Finally, let us assume that the alphabet is $\Sigma = \{a_1, \ldots, a_n\}$ and let $j_1, \ldots, j_n \in \mathbb{N}$ be large integers. If we define $k = \sum_{i=1}^n |\mathrm{CQ}(a_i^{j_i})|_{a_i}$, then Lemma 15 brings $k = \sum_{i=1}^n (|o|_{a_i} + r) = |o| + |\Sigma| \cdot r \geq |o| + r$. So we can plug k into Proposition 14 and get a string $w = \mathrm{CQ}\left((a_1 \ldots a_n)^k\right) \in B_r^{max}(o)$. Moreover, we have $|w| = |o| + r$ and $k = |o| + |\Sigma| \cdot r$. So, we deduce that the radius is $r = (k - |w|)/(|\Sigma| - 1)$.

4.2 An Algorithm to Learn Balls from Correction Queries

Let us summarize, by assuming that $\Sigma = \{a_1, \ldots, a_n\}$ and that the target is the ball $B_r(o)$. (1) For each letter a_i, the Learner asks for the correction of a_i^j where j is sufficiently large to get a correction whose length is smaller than j; (2) the Learner sets $k = \sum_{i=1}^{n} |\text{CQ}(a_i^{j_i})|_{a_i}$ and gets the correction w of the string $v = (a_1 \ldots a_n)^k$; (3) from k and $|w|$, he deduces r; (4) he uses EXTRACT_CENTRE on w and r, and he gets o. In other words, he is able to guess the balls with correction queries (see Algorithm IDF_BALL and Proposition 16).

Algorithm 2 IDF_BALL

Require: The alphabet $\Sigma = \{a_1, \ldots, a_n\}$
Ensure: The representation (o, r) of the target ball $B_r(o)$
1: $j \leftarrow 1; k \leftarrow 0$
2: **for** $i = 1$ **to** n **do**
3: **while** $\text{CQ}(a_i^j) = \text{YES}$ **or else** $|\text{CQ}(a_i^j)| \geq j$ **do**
4: $j \leftarrow 2 \cdot j$
5: **end while**
6: $k \leftarrow k + |\text{CQ}(a_i^j)|_{a_i}$
7: **end for**
8: $w \leftarrow \text{CQ}((a_1 a_2 \ldots a_n)^k)$
9: $r \leftarrow (k - |w|)/(|\Sigma| - 1)$
10: $o \leftarrow \text{EXTRACT\_CENTRE}(w, r)$
11: **return** (o, r)

For instance, consider the ball $B_2(11)$ defined over $\Sigma = \{0, 1\}$. IDF_BALL begins by looking for the corrections of 0^j and 1^j with a sufficiently large j. We might observe: $\text{CQ}(0) = \text{YES}$, $\text{CQ}(0^2) = \text{YES}$, $\text{CQ}(0^4) = 0011$, $\text{CQ}(0^8) = 0110$, $\text{CQ}(1^8) = 1111$. So $k = |0110|_0 + |1111|_1 = 2 + 4 = 6$. Then $\text{CQ}\left((01)^6\right) = \text{CQ}(010101010101) = 0110$, for instance, so $r = (6-4)/(2-1) = 2$. Finally, $\text{EXTRACT\_CENTRE}(0110, 2)$ returns 11. So the algorithm returns $(11, 2)$, which is the representation of the target ball.

Proposition 16 *Given any fixed ball $B_r(o)$, the Algorithm* IDF_BALL *returns the representation (o, r) using $O(|\Sigma| + |o| + r)$ correction queries.*

Proof The correction of IDF_BALL is clear. Concerning the number of queries, the corrections of all the strings a_i^j require $O(|\Sigma| + \log(|o| + r))$ correction queries (lines 2-5). Indeed, $O(\log(|o| + r))$ queries are necessary to get long enough corrections, plus one query per letter, thus $|\Sigma|$ queries. Then EXTRACT_CENTRE needs $O(|o| + r)$ correction queries (line 8) to find the centre, by Proposition 13. So the total amount of queries is $O(|\Sigma| + |o| + r)$. ■

4.3 Only the Good Balls are Learnable with IDF_BALL

We now have an algorithm that guesses all the balls $B_r(o)$ with $O(|\Sigma| + |o| + r)$ correction queries. Is this result relevant?

In Section 3, we have decided to represent every ball $B_r(o)$ by the pair (o,r). The size of this representation, $|o|+\log r$, is basically related to the number of bits needed to encode this representation. Notice that $|o|+r$ *is not* a correct measure of the size: this would correspond to an encoding in base 1 of the radius r, which is not considered reasonable (Garey and Johnson, 1979). Therefore, the number of correction queries used by IDF_BALL is exponential in $\log r$. In consequence, if r is too large with respect to $|o|$, (e.g., $r > 2^{|o|}$), then our algorithm is not able to identify efficiently the target ball.

In order to avoid this problem, we introduce the following definition that allows us to rewrite Proposition 16 in a more relevant way:

Definition 17 (Good Balls)

- *Let $q()$ be any fixed polynomial. We say that a ball $B_r(o)$ is* $q()$-good *if $r \leq q(|o|)$.*
- *We say that $B_r(o)$ is* very good *if $r \leq |o|$.*

A very good ball is thus a $q()$-good ball for the polynomial $q(x) = x$.

Then, Proposition 16 yields:

Theorem 18

- *Let $q()$ be any fixed polynomial. The set of all $q()$-good balls $B_r(o)$ is identifiable with an algorithm that uses $O(|\Sigma| + |o| + q(|o|))$ correction queries and a polynomial amount of time.*
- *The set of all very good balls is identifiable with a linear number $O(|\Sigma| + |o|)$ of correction queries and a polynomial amount of time.*

Finally, the reader may wonder if a better learnability result could be established, which would include the huge balls. Unfortunately, there is not a unique answer to this question. On one hand, if the number of correction queries authorized to learn can also depend on the length of the longest correction provided by the Oracle during a run of IDF_BALL, then the answer is positive: all the balls are learnable. On the other hand, in de la Higuera et al. (2008), it has been proved that the set of all the balls was not polynomially identifiable in the limit, nor in most relevant online learning paradigms, whereas positive results were established for the good balls in most paradigms. From this point of view, Theorem 18 is satisfying.

5. Learning in a Realistic Environment

The setting of learning with queries itself occurs in many circumstances: when a human being is asked to provide data for a learning program, an alternative to have the human expert labeling huge quantities of data can be to have the learning system interact with the human expert, who then only labels those items required. Nevertheless, assuming that the Oracle is a human expert introduces new constraints. On one hand, asking billions of queries is unacceptable: there is no chance to get enough answers in reasonable time. So the learning algorithm should aim at minimizing the number of queries even if we must pay for it with a worse time complexity. On the other hand, a human (or even the Web) is fallible. Therefore, the learning algorithm should aim at learning functions or languages that are robust from corrections that may not be ideal, thus approximative.

These issues are discussed in de la Higuera (2006). Some examples of this alternative approach (imperfect Oracle) are: system SQUIRREL, which makes use of queries to a human expert to allow

wrapper induction (Carme et al., 2007); learning test sets (Bréhélin et al., 2001) and testing hardware (Hagerer et al., 2002), where the actual electronic device can be physically tested by entering a sequence, and the device will then be able to answer a membership query (note that in that setting equivalence queries will be usually simulated by sampling). Another typical question occurs when learning some non intelligible function, defined perhaps by a complex kernel (Clark et al., 2006) or neural networks (Giles et al., 2001): a representation of these complex functions in another setting can be obtained if we use the complex function as an Oracle to learn from.

5.1 Faced with a Fallible Oracle

The algorithm IDF_BALL has been designed in an ideal setting, where we have assumed that the Oracle was a perfect machine: her answers were so precise that we could scrupulously characterize them (see Lemma 10). However, as described in the introduction, in practice, an Oracle is often an expert, thus a human being, or is simulated through sampling. In such settings, our assumption is no longer correct. Indeed, computing the correction of $(101)^{127}$ *w.r.t.* the ball $B_{217}((1011)^{32})$ is probably out of the cognitive capacities of any human being. So our algorithm should not believe unwisely the answers he gets since they can be approximate. In this section, we would like to show, with a series of experiments, that our algorithm withstands such approximate (that is to say, inaccurate, noisy) answers.

5.1.1 DESIGNING THE APPROXIMATE ORACLE

We want here to design an approximate Oracle that might look like a human being. So let us consider a string w and a ball $B_r(o)$. Let $\mathrm{CQ}_h(w)$ denote the answer of the approximate Oracle, and $\mathrm{CQ}(w)$ the answer that would be returned by a perfect Oracle (as before).

Firstly, we assume that an expert can easily determine whether an example fulfills a concept or not, thus here, whether w belongs to $B_r(o)$ or not. So we assume that if $\mathrm{CQ}(w) = \mathrm{YES}$, then $\mathrm{CQ}_h(w) = \mathrm{YES}$. Secondly, what is really hard for the expert is to *compute* the best correction of w when $w \notin B_r(o)$, and more precisely, a string of the ball that is *as close to w as possible*. Again, $\mathrm{CQ}_h(w)$ will probably be inside the ball rather than on its frontier.

Staying a step ahead, let $X = d(w, \mathrm{CQ}_h(w)) - d(w, \mathrm{CQ}(w))$ measure the distance between an approximate correction and a perfect one. Intuitively, an approximate but strong Oracle will often provide corrections such that $X = 0$, sometimes $X = 1$ and rarely $X \geq 2 \ldots$ To formalize this idea, we introduce a confidence parameter $0 < p \leq 1$, called the *accuracy level of the Oracle*, that translates the quality of her answers, and use a geometric distribution: $Pr(X = k) = (1-p)^k p$, for all $k \in \mathbb{N}$.

Therefore, with probability $(1-p)^k p$, the correction $\mathrm{CQ}_h(w)$ of a string w will be in the target ball, at distance k of $\mathrm{CQ}(w)$. Basically, we get $E(X) = (1/p) - 1$. So when the Oracle is very accurate, say $p = 0.8$, then the average distance between an approximate and a perfect correction is low (0.25). Conversely, an expert with limited computation capacities, say $p = 0.4$, will often provide inaccurate corrections, at distance 1.5 on average.

Our model of approximate Oracle is simple. For instance, we do not suppose that she has any memory, thus by submitting twice every string w, we would probably get 2 different corrections that could be used to correct the corrections! We want here to study the resistance of IDF_BALL to approximate answers, not to design the best possible algorithm, able to beat the approximate Oracle. So from this standpoint, our basic model is sufficient.

5.1.2 BEHAVIOR OF THE ALGORITHM FACED WITH AN APPROXIMATE ORACLE

Following Theorem 18, IDF_BALL systematically guesses the target ball with the help of a perfect Oracle. But of course, he is sometimes going to fail in front of an approximate Oracle. So, in order to assess the resistance of IDF_BALL to approximate corrections, we conduct the following experiment. We randomly choose a set of 100 balls $B_r(o)$ such that $|o| + r = 200$. More precisely, we make the radius r vary between 10 and 190 by step of 20, and randomly choose 10 centres o of length $200 - r$ for each radius. Then, for every accuracy level $0.5 \leq p \leq 1$, we ask IDF_BALL to learn all of them and we compute the percentage of balls he is able to retrieve, that we call the *precision of the algorithm*. We show the result in Figure 4. We notice that IDF_BALL is able to identify about 75% of the balls faced with an accuracy level of $p = 0.9$. Of course, as one can expect, with lower levels of accuracy, his performances quickly drop (15% for $p = 0.5$).

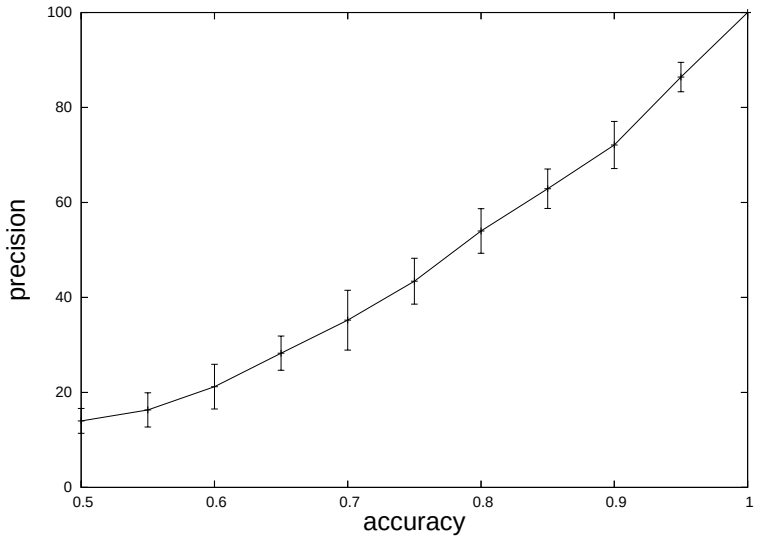


Figure 4: Precision of IDF_BALL faced with an approximate Oracle in function of the accuracy level p. Each point is assessed on 100 balls.

We also show, in Figure 5, the average distances between the centres of the target balls and the centres of the the learnt balls when he fails to retrieve them. We observe that these distances are not that important: even with an accuracy level of $p = 0.5$, the difference is less than 3. The last curve in Figure 6 is the difference between the radii, that basically follow the same trend.

5.1.3 IMPROVING THE PRECISION WITH *a posteriori* HEURISTICS

We have seen that IDF_BALL was able to assimilate the approximations of the Oracle up to a certain level of accuracy. Moreover, the centres and the radii returned by the algorithm are generally not far from the target. Therefore, it is reasonable to think that we could improve the precision by exploring the neighborhood of the learnt centre, using local edit modifications. This kind of approaches has been pioneered by Kohonen (1985) and is surveyed in Martínez-Hinarejos et al. (2000).

Suppose that the learnt ball is $B_k(u)$ and let Q be the set of all the corrections returned by the Oracle during the process of IDF_BALL. The heuristics is composed of two steps:

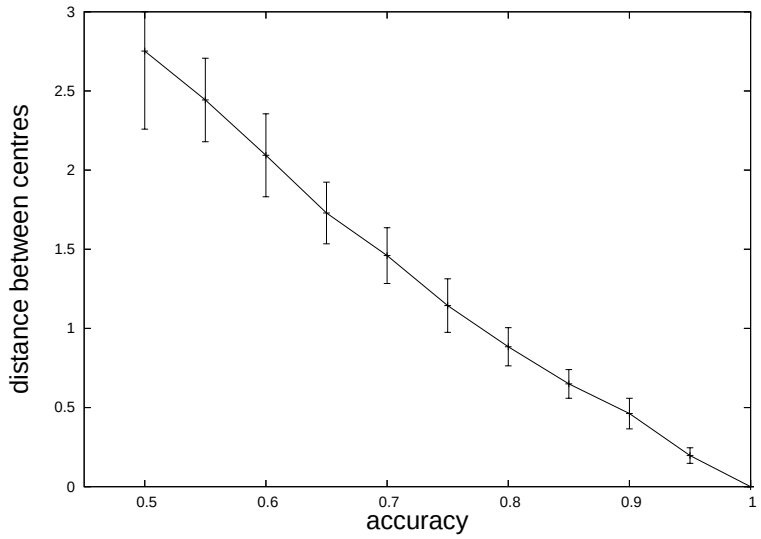


Figure 5: Average distances (and standard deviation) between the centres of the target balls and the centres of the learnt balls, when IDF_BALL fails in retrieving them.

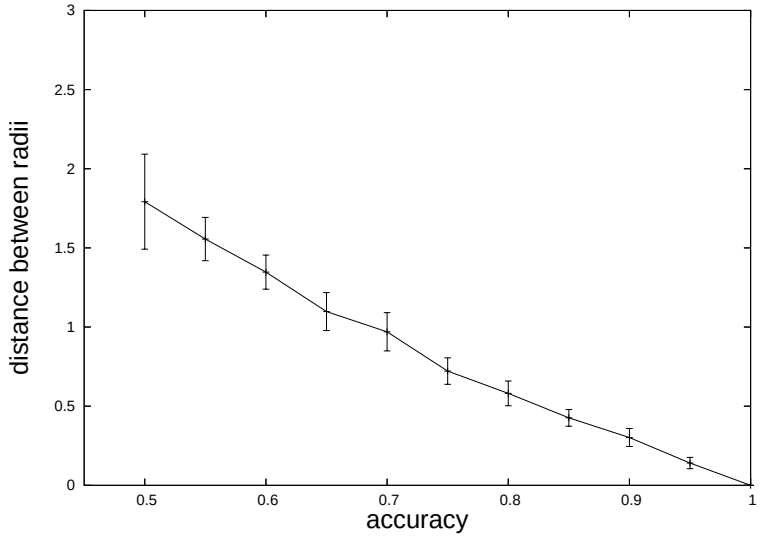


Figure 6: Average difference (and standard deviation) between the radii of the target balls and the radii of the learnt balls, when IDF_BALL fails in retrieving them.

1. We test each neighbor u' (at distance 1) of the learnt centre u and examine if it is a better centre with respect to Q, that is to say, if there exists $k' \in \mathbb{N}$ such that $k' < k$ and $Q \subseteq B_{k'}(u')$. Then we keep the representations (u', k') of the smallest balls that contain all the corrections seen so far.

2. From this set, we select all the pairs (u', k') that maximize the number of corrections (of Q) at distance k', in such a way as to get the maximum number of corrections on the border of the new ball. Then we randomly choose and return one of them.

This heuristics will be very good each time u is at distance 1 from the target centre. But as soon as this distance grows, IDF_BALL will fail again. In order to enhance the one-step heuristics, we iterate the process and design a second until-convergence heuristics by repeating the local search described above, until the size of the ball cannot decrease anymore.

In order to show that the balls learnt by IDF_BALL can be corrected *a posteriori*, we compare, in a series of experiments, the precision of the algorithm without any post-treatment, with the one-step heuristics and with the until-convergence heuristics. We fix $|o| + r = 200$ and make the radius vary from 10 to 190. For each radius, we randomly draw 50 centres of length $200 - r$. Then, we make the accuracy level vary from 0.5 to 1. For each pair (accuracy, radius), we ask IDF_BALL to retrieve the 50 balls and note the precision. In order to be able to reduce the variance due to the approximations of the Oracle, we repeat the experiment 10 times using the same set of balls and finally plot the average precisions in Figure 7.

We can remark that whatever the accuracy level, using the until-convergence heuristics is never worse than the one-step heuristics, which is never worse than no post-treatment at all. However, our heuristics do not always improve the precision of the algorithm: this depends on the ratio between the radius of the target ball and the length of its centre. In order to detail this, we have extracted two transverse sections, shown in Figures 8 and 9, where we fix the radii.

Figure 8 describes the precision of IDF_BALL for target balls such that $r = 170$ and $|o| = 30$. In this case, we gain little using the heuristics. This is probably due to the fact that the size of the set Q, which is used to control the heuristics, is incomparably smaller than the volume of such balls. In other words, the heuristics are not sufficiently guided by Q towards the targets, because Q is not informative enough.

On the other hand, Figure 9 describes the precision for target balls such that $r = 10$ and $|o| = 190$. Basically, our heuristics outperform the precision with respect to the algorithm without any post-treatment, whatever the accuracy level of the Oracle. Moreover, the benefit is all the more important as the accuracy level is bad. For instance, when $p = 0.6$, the until-convergence heuristics is able to dramatically boost the precision from 12% to 86%.

So in this setting, with no further enhancement, IDF_BALL produces balls that are so close to the targets that they can easily be improved using only basic local modifications.

5.2 Using Less Correction Queries

We have seen that the good balls were identifiable with $O(|\Sigma| + |o| + r)$ correction queries. However, as discussed in the introduction, such a number of queries is excessive if the Oracle is a human being. Moreover, if the reader thinks of what happens in the plane (see Figure 2), then very few queries are needed to identify the disks. Hence, our result might seem to be a bit disappointing.

If one takes a closer look at IDF_BALL, one can notice that the first part of the identification, that is to say, the search for a string of $B_r^{max}(o)$, is done with $O(|\Sigma| + \log(|o| + r))$ correction queries

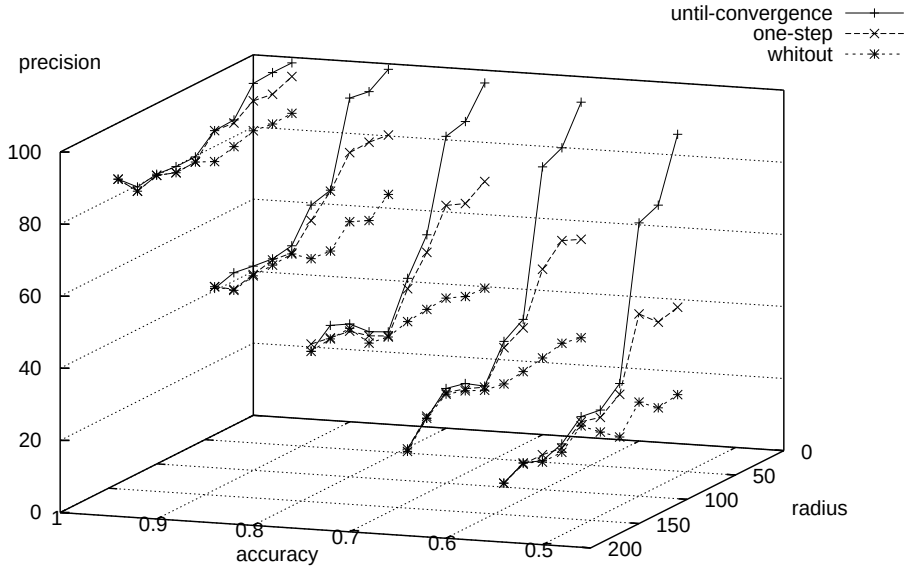


Figure 7: Precision of IDF_BALL with and without heuristics in function of accuracy and radius when $|o| + r = 200$. For each pair (accuracy, radius), we compute the average over 50 balls.

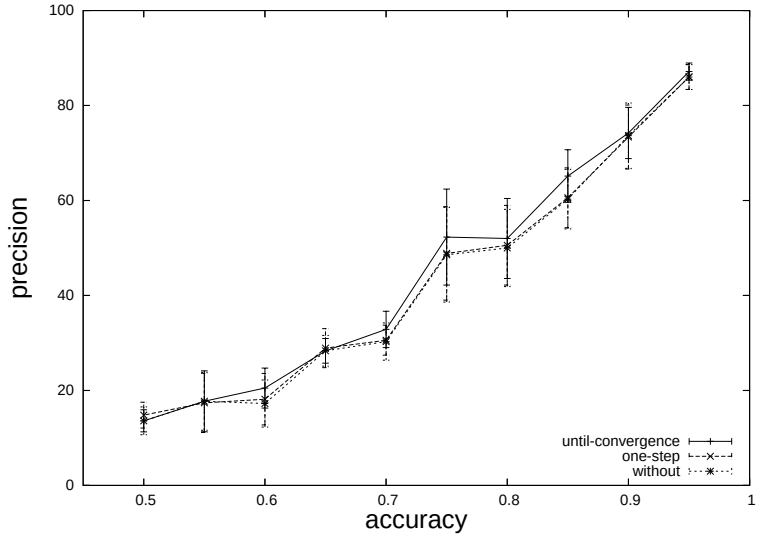


Figure 8: Precision of IDF_BALL when $|o| + r = 200$ for $r = 170$. For each accuracy, we compute the average over 50 balls. We run the experiment 10 times in order to reduce the variance.

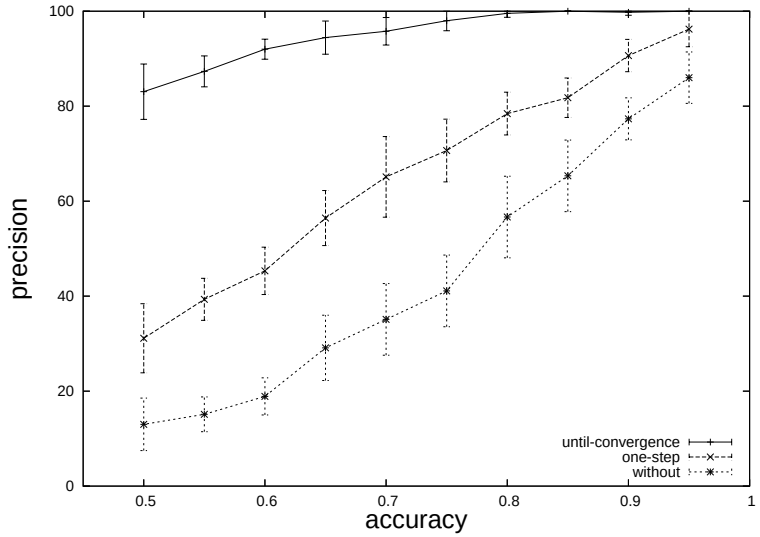


Figure 9: Precision of IDF_BALL when $|o| + r = 200$ for $r = 10$. For each accuracy, we compute the average over 50 balls. We run the experiment 10 times in order to reduce the variance.

(thus, a logarithmic number). What is really expensive is to find the centre of the ball using the function EXTRACT_CENTRE. We are going to show below that this function can be eliminated from the learning stage, and thus, that the complexity can be dramatically reduced, but in a slightly different setting.

For reasons that we shall develop later, we now suppose that the alphabet has at least three letters: $\Sigma = \{a_1, \ldots, a_n\}$ with $n \geq 3$.

5.2.1 THE USE OF A WEIGHTED EDIT DISTANCE

Up to now, we have considered the standard edit distance defined by Levenshtein (1965). However, for practical reasons, people often use variants of this definition where the edit operations are weighted. In this case, every derivation $w \xrightarrow{k} w'$ has a weight which is the sum of the weights of the edit operations along the derivation. Then the *weighted edit distance* $d(w, w')$ is the minimum weight of every derivation transforming w into w'. Clearly, if the weight of all the edit operations is 1, then we get the standard edit distance.

The different combinations of weights will impose alternative algorithms when using correction queries. As we aim at showing that the number of correction queries can be dramatically reduced, we assume that:

1. the weight of every insertion and every deletion is 1 (as before),

2. the weight of every substitution is an irrational number ρ such that $0 < \rho < 1$.

For instance, the weight of the substitution could be $\rho = \frac{\pi}{4} \simeq 0.7854$.

It is worth noting that the low cost of a substitution operation is usual from a linguistic point of view. For instance, works on Phonology make this assumption in order to enforce the alignment of phonetically similar segments (Albright and Hayes, 2003).

Nevertheless, the fact that ρ is not rational may be confusing for the reader. Actually, from the Learner standpoint, we will see that he never needs to compute the weighted edit distance (that is probably not the case of the Oracle). So the fact that ρ is not a fraction will not be a problem.

We can show that this set of weights induces an edit distance that can be computed using dynamic programming.[3] Moreover, Proposition 2, stating that (1) $d(w,w') \geq \big||w|-|w'|\big|$ and (2) $d(w,w') = \big||w|-|w'|\big|$ *iff* ($w \preceq w'$ or $w' \preceq w$), still holds, because the weight of the insertions and deletions is 1. Finally, the fact that ρ is irrational allows us to establish strong properties:

Proposition 19 *Let $o,w,w' \in \Sigma^*$ be three strings. The following statements are equivalent:*

1. *There exists a derivation of minimum weight from w to w' that uses $x \in \mathbb{N}$ insertions and deletions, and $y \in \mathbb{N}$ substitutions;*
2. *$d(w,w') = x + \rho y$;*
3. *All the derivations of minimum weight from w to w' use $x \in \mathbb{N}$ insertions and deletions, and $y \in \mathbb{N}$ substitutions.*

In consequence, if $d(o,w) = d(o,w')$, then all the derivations from o to w and from o to w' use the same number of insertions and deletions, and the same number of substitutions.

Proof

- 3. $\Longrightarrow$ 1.: straightforward.
- 1. $\Longrightarrow$ 2.: since the weight of the insertions and deletions is 1, and the weight of the substitutions is ρ, and the derivation has a minimum weight, we get $d(w,w') = x + \rho y$.
- 2. $\Longrightarrow$ 3.: consider another derivation from w to w' of minimum weight that uses $x' \in \mathbb{N}$ insertions and deletions, and $y' \in \mathbb{N}$ substitutions. Then we get $d(w,w') = x' + \rho y' = x + \rho y$, so $x - x' = \rho(y' - y)$. As ρ is irrational and x, x', y, y' are integers, we deduce that $y' - y = 0$ and $x - x' = 0$, thus $x' = x$ and $y' = y$.

■

Of course, this result would not hold if ρ was a rational number, for instance $\rho = \frac{1}{2}$, because two substitutions would have the same weight as one insertion, which might induce two very different derivations of minimum weight between two strings.

5.2.2 THE NEW GOOD BALLS AND CORRECTIONS

Basically, changing the edit distance also changes the balls. For instance, using the standard edit distance, we get $B_1(011) = \{01, 11, 001, 010, 011, 111, 0011, 0101, 0110, 1011, 0111\}$. But the use of the weighted edit distance with $\rho = \frac{\sqrt{2}}{4} \simeq 0.3536$ adds $\{000, 101, 110\}$ as new strings.

3. Indeed, its restriction to $\Sigma \cup \{\lambda\}$ is a distance, so following Crochemore et al. (2007), the standard dynamic programming algorithm of Wagner and Fisher (1974) can be used to compute the weighted edit distance over Σ^*.

The reader may also wonder whether the radius of the balls should still be an integer or not. Actually, we shall not consider balls whose radius is not an integer, because otherwise, the balls $B_r(o)$ and $B_{r+\frac{\rho}{2}}(o)$ might represent the same set. In other words, Theorem 5, that states the uniqueness of the representation, would not hold anymore. Conversely, if we only consider balls with an integer radius, then the reader can check that Theorem 5 still holds (because Proposition 2 still holds).

Concerning the corrections, their properties become more intricate due to the weights. In particular, Lemma 10 was stating that the set of possible corrections of any string $v \notin B_r(o)$ was exactly $\{u \in \Sigma^* : d(o,u) = r \text{ and } d(u,v) = d(o,v) - r\}$. This result does not hold anymore. Indeed, consider the ball $B_1(011)$ when $\rho = \frac{\sqrt{2}}{4}$. Then any correction u of the string 100 is in $\{000, 101, 110\}$. Basically, $d(011,u) = 2\rho < 1$ and $d(u,100) = \rho > d(011,100) - 1 = 3\rho - 1$. In other words, a correction is not necessarily on the circle that delimits the ball.

Nevertheless, we get a more sophisticated result that characterizes the set of all the possible corrections:

Lemma 20 *Let $B_r(o)$ be a ball and $v \notin B_r(o)$. Given any $\alpha \in \mathbb{R}$, we define*

$$C_\alpha = \{u \in \Sigma^* : d(o,u) = \alpha \text{ and } d(u,v) = d(o,v) - \alpha\}.$$

All the nonempty C_α define concentric arcs of circles of strings around the centre o. Let α_0 be the radius of biggest one inside the ball of strings:

$$\alpha_0 = \max_{0 \leq \alpha \leq r} \{\alpha : C_\alpha \neq \emptyset\}.$$

Then the set of possible corrections of v is exactly C_{α_0}.

Proof The proof is the same as that of Lemma 10, except that W is replaced by C_{α_0} and r is replaced by α_0. The key point is that W could be empty with the weighted edit distance whereas C_{α_0} cannot, by definition. ■

5.2.3 THE BORDERLINE STRINGS OF MAXIMUM LENGTH

Let us tackle the problem of learning the balls. As in Section 4, we study the longest strings of $B_r(o)$ since they are very informative. Indeed, we are going to show as in Lemma 15, that if one asks for the correction w of a string made of a lot of 0's, then $|w|_0 = |o|_0 + r$. In addition, in our setting, we also get $w \in B_r^{max}(o)$ directly. Nevertheless, we must pay for it by assuming that we know the polynomial $q()$ for which $B_r(o)$ is a good ball.

Lemma 21 *Let $q()$ be a fixed polynomial with coefficients in $\mathbb{N}$. Consider the $q()$-good ball $B_r(o)$, a letter $a \in \Sigma$ and an integer $j \in \mathbb{N}$ such that $a^j \notin B_r(o)$. Let $u = \mathrm{CQ}(a^j)$ and $v = \mathrm{CQ}(a^{j+q(j)})$. If $|u| < j$, then $v \in B_r^{max}(o)$ and $|v|_a = |o|_a + r$.*

This subsection aims at proving this lemma, using two intermediate results:

Proposition 22 *Consider the ball $B_r(o)$, a letter $a \in \Sigma$ and an integer $j \in \mathbb{N}$ such that $a^j \notin B_r(o)$. Let $u = \mathrm{CQ}(a^j)$. If $|u| < j$, then $|o| < j$.*

Proof Let us show that $|o| \leq |u|$; as $|u| < j$, we shall get the result. Hence, suppose that $|u| < |o|$ and consider a rewriting derivation $o \xrightarrow{k} u$ of minimum weight $d(o,u) = x + \rho y$. Since $|o| > |u|$, there is at least one deletion along this derivation. Suppose that, instead of deleting a letter, we substitute it by an a and do not change anything else. This leads us to a new derivation $o \xrightarrow{k} u'$ (or $o \xrightarrow{k-1} u'$ if the deleted letter was an a) with $|u'| = |u| + 1$ and $|u'|_a = |u|_a + 1$. Moreover, $d(o,u') \leq (x-1) + \rho(y+1) = d(o,u) - 1 + \rho$. Since $d(o,u) \leq r$, we deduce that $d(o,u') < r$, thus $u' \in B_r(o)$. Finally, as $|u| < j$, we get $|u'| \leq j$, so only substitutions and insertions are necessary to compute both $d(u,a^j)$ and $d(u',a^j)$. More precisely, we have $d(u',a^j) = (j - |u'|) + \rho(|u'| - |u'|_a) = (j - |u| - 1) + \rho(|u| - |u|_a) = d(u,a^j) - 1$, thus $d(u',a^j) < d(u,a^j)$. As $u' \in B_r(o)$, u cannot be a correction of a^j, which is a contradiction. So $|u| \geq |o|$, thus $|o| < j$. ■

Proposition 23 *Consider the ball $B_r(o)$, a letter $a \in \Sigma$ and an integer $\ell \in \mathbb{N}$ such that $a^\ell \notin B_r(o)$. Let $v = \mathrm{CQ}(a^\ell)$. If $|o| + r < \ell$, then $v \in B_r^{max}(o)$ and $|v|_a = |o|_a + r$.*

Proof As $|o| + r < \ell$, we have $|o| < \ell$, so the computation of $d(o,a^\ell)$ necessarily uses $\ell - |o|$ insertions of a's and $|o| - |o|_a$ substitutions by a's. Let us define a reference derivation from o to a^ℓ, where the $\ell - |o|$ insertions are performed first at the beginning of o, and then the $|o| - |o|_a$ substitutions by a's in o: $o \xrightarrow{\ell-|o|} a^{\ell-|o|}o \xrightarrow{|o|-|o|_a} a^{\ell-|o|}a^{|o|} = a^\ell$. As $\ell - |o| > r$, the string $a^{\ell-|o|}o$ is not in the ball, so this derivation passes through the string $a^r o$ before reaching $a^{\ell-|o|}o$. In other words, the reference derivation looks as follows: $o \xrightarrow{r} a^r o \xrightarrow{\ell-|o|-r} a^{\ell-|o|}o \xrightarrow{|o|-|o|_a} a^\ell$. Now consider Lemma 20. Basically, $d(o,a^r o) = r$ and $d(a^r o,a^\ell) = \ell - |o| - r + \rho(|o| - |o|_a) = d(o,a^\ell) - r$, so $C_r \neq \emptyset$. Therefore, as $v = \mathrm{CQ}(a^\ell)$, we deduce that $d(o,v) = r$ and $d(v,a^\ell) = d(o,a^\ell) - r$. We claim that only insertions of a's can occur along the derivation $o \xrightarrow{r} v$. Indeed, we have $d(o,v) = r \in \mathbb{N}$, so by Proposition 19, no substitution occurs. Moreover, no deletion occurs since any minimal derivation from o to a^ℓ only uses insertions of a's and substitutions by a's. In consequence, $v \in B_r^{max}(o)$ and $|v|_a = |o|_a + r$. ■

Proof [of Lemma 21] By Proposition 22, we get $|o| < j$. Then we have $|o| + r \leq |o| + q(|o|)$. Moreover, as $|o| < j$ and all the coefficients of $q()$ are in $\mathbb{N}$, we deduce that $|o| + r < j + q(j)$. So plugging $\ell = j + q(j)$ in Proposition 23 yields the result. ■

5.2.4 LEARNING THE BALLS LOGARITHMICALLY

As a consequence of Lemma 21, the correction of a long string of 0's leads to a string of $B_r^{max}(o)$. But we get more properties, if the alphabet has at least three letters, say $0,1,2\ldots$ Indeed, let $u_0 = \mathrm{CQ}(0^j)$ with $|u_0| < j$, and $v_0 = \mathrm{CQ}(0^{j+q(j)})$. Thanks to Lemma 21, v_0 is obtained from o with r insertions of 0's. So all the letters in v_0, but the occurrences of 0, are those of o and appear in the correct order.

More formally, let E_a be the function that erases every occurrence of any letter $a \in \Sigma$ in a string:

1. $E_a(\lambda) = \lambda$,
2. $E_a(a.z) = E_a(z)$, and

3. $E_a(b.z) = b.E_a(z)$, for all $b \neq a$.

Then, for every $a \in \Sigma$, $E_a(v_a) = E_a(o)$.

For instance, consider the ball $B_1(o)$ with $o = 10302$. If the corrections of the strings 0^ℓ, 1^ℓ and 2^ℓ (with ℓ big enough) are $v_0 = 103020$, $v_1 = 101302$ and $v_2 = 103202$ respectively, then $E_0(v_0) = E_0(o) = 132$, $E_1(v_1) = E_1(o) = 0302$ and $E_2(v_2) = E_2(o) = 1030$.

Furthermore, we can easily deduce o by *aligning* the strings $E_0(o)$ and $E_1(o)$ and $E_2(o)$:

$E_0(o)$	1	·	3	·	2
$E_1(o)$	·	0	3	0	2
$E_2(o)$	1	0	3	0	·
o	1	0	3	0	2

This procedure does not use any new correction query and runs in time $O(|o|)$ which is clearly more efficient than EXTRACT_CENTRE. Notice that if $|\Sigma| > 3$, we only need three corrections to align and deduce the center. So we finally obtain Algorithm 3 and Theorem 24.

Algorithm 3 IDF_WEIGHTED_BALLS

Require: The alphabet $\Sigma = \{a_1, \ldots, a_n\}$ with $n \geq 3$, and the polynomial $q()$
Ensure: The representation (o, r) of the target $q()$-good ball $B_r(o)$

$j \leftarrow 1$
for $i = 1$ to 3 **do**
  while $\text{CQ}(a_i^j) = \text{YES}$ **or else** $|\text{CQ}(a_i^j)| \geq j$ **do**
    $j \leftarrow 2 \cdot j$
  end while
  $v_i \leftarrow \text{CQ}\left(a_i^{j+q(j)}\right)$
  $e_i \leftarrow E_{a_i}(v_i)$
end for
$o \leftarrow \text{ALIGN}(e_1, e_2, e_3)$
$r \leftarrow |v_1| - |o|$
return (o, r)

Theorem 24 *Assume* $|\Sigma| \geq 3$.

- *Let $q()$ be any fixed polynomial with coefficients in* $\mathbb{N}$. *The set of all $q()$-good balls $B_r(o)$ is identifiable with an algorithm that uses $O(\log(|o| + q(|o|)))$ correction queries and a polynomial amount of time.*
- *The set of all very good balls is identifiable with a logarithmic number $O(\log |o|)$ of correction queries and a polynomial amount of time.*

Therefore, assuming that the weight of the substitutions is an irrational < 1 allows us to reduce dramatically the complexity of the learning stage. Of course, this gain is not possible with all weighted distances, which leaves room for further research. Moreover, if the Learner does not know the polynomial $q()$, we believe that learning is still possible.

6. Discussion and Conclusion

In this work, we have used correction queries to learn a particular class of languages from an Oracle. The intended setting is that of an inexact Oracle, and experiments show that the proposed algorithm can learn a language sufficiently close to the target for simple local modifications (with no extra queries). In order to do this, the languages we consider are good balls of strings defined with the edit distance. Studying them allowed us to catch a glimpse of the geometry of sets of strings, which is very different from the Euclidean geometry. A number of questions and research directions are left open by this work.

A first question concerns the distance we use. We have chosen to work with the unitary edit distance, but in many applications, the edit operations can have different weights. Preliminary work has allowed us to notice that the geometry of sets of strings, thus the algorithmics, could change considerably depending on the sorts of weights we used: with the substitutions costing less than the other two operations, a much faster algorithm exists, requiring only $O(\log(|o|+r))$ correction queries. Alternative conditions over the weights require new interesting learning algorithms.

A second question concerns the inaccuracy model we are using: as noticed in Section 5.1, with the current model it would be possible to repeat the same query various times, getting different corrections, but possibly being able, through some majority vote scheme, to get the adequate correction with very little extra cost. Just asking for *persistent* corrections is not enough to solve this problem: a good model should require that if one queries from a close enough string (a^{999} instead of a^{1000}) then the corrections should also remain close. Topologically, we would expect the Oracle to be k-Lipschitz continuous (with $0 < k < 1$).

A third more challenging problem then arises: our choice here was to learn supposing the Oracle was exact, and correcting later. But a more direct approach might be better, by taking into account the inexactitude of the Oracle when interpreting the correction.

Acknowledgments

The authors wish to thank Jose Oncina for his help in proving Theorem 5, Rémi Eyraud for fruitful discussions about this paper, Dana Angluin for constructive comments and Baptiste Gorin for his helpful pointers towards the Mohr-Mascheroni constructions. We would also like to thank the anonymous referees that have carefully read this manuscript and allowed us to improve the results based on the weighted edit distance. Their remarks led us to formulate Conjecture 4 that was discussed with Ron Greensberg, Borivoj Melichar, Klaus Schultz and Stoyan Mihov. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2006-216886, and by a Marie Curie International Fellowship within the 6th European Community Framework Programme. This publication only reflects the authors' views.

References

A. Albright and B. Hayes. Rules vs. analogy in English past tenses: A computational/experimental study. *Cognition*, 90:119–161, 2003.

J.-C. Amengual and P. Dupont. Smoothing probabilistic automata: An error-correcting approach. In *Proc. of the 5th International Colloquium in Grammatical Inference (ICGI'00)*, pages 51–64.

LNAI 1891, 2000.

J.-C. Amengual, A. Sanchis, E. Vidal, and J.-M. Benedí. Language simplification through error-correcting and grammatical inference techniques. *Machine Learning Journal*, 44(1-2):143–159, 2001.

D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987a.

D. Angluin. Queries revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39: 337–350, 1978.

D. Angluin. Queries and concept learning. *Machine Learning Journal*, 2(4):319–342, 1987b.

D. Angluin and M. Kharitonov. When won't membership queries help? *Journal of Computer and System Sciences*, 50(2):336–355, 1995.

L. Becerra-Bonache. *On the Learnability of Mildly Context-Sensitive Languages using Positive Data and Correction Queries*. PhD thesis, Rovira i Virgili University, Tarragona, 2006.

L. Becerra-Bonache and T. Yokomori. Learning mild context-sensitiveness: Towards understanding children's language learning. In *Proc. of the 7th International Colloquium in Grammatical Inference (ICGI'04)*, pages 53–64. LNAI 3264, 2004.

L. Becerra-Bonache, A. H. Dediu, and C. Tirnauca. Learning DFA from correction and equivalence queries. In *Proc. of the 8th International Colloquium in Grammatical Inference (ICGI'06)*, pages 281–292. LNAI 4201, 2006.

L. Bréhélin, O. Gascuel, and G. Caraux. Hidden Markov models with patterns to learn boolean vector sequences and application to the built-in self-test for integrated circuits. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):997–1008, 2001.

J. Carme, R. Gilleron, A. Lemay, and J. Niehren. Interactive learning of node selecting tree transducers. *Machine Learning*, 66(1):33–67, 2007.

E. Chávez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

N. Chomsky. *Syntactic Structure*. Mouton, 1957.

A. Clark, C. Costa Florêncio, and C. Watkins. Languages as hyperplanes: Grammatical inference with string kernels. In *Proc. of the 17th European Conference on Machine Learning (ECML'06)*, pages 90–101. LNCS 4212, 2006.

F. Coste, K. Lang, and B. A. Pearlmutter. The Gowachin automata learning competition, 1998. URL `http://www.irisa.fr/Gowachin/`.

M. Crochemore, C. Hancart, and T. Lecroq. *Algorithms on Strings*. Cambridge University Press, 2007.

C. de la Higuera. Data complexity issues in grammatical inference. In M. Basu and T. K. Ho, editors, *Data Complexity in Pattern Recognition*, pages 153–172. Springer-Verlag, 2006.

C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27: 125–138, 1997.

C. de la Higuera and F. Casacuberta. Topology of strings: Median string is NP-complete. *Theoretical Computer Science*, 230:39–48, 2000.

C. de la Higuera, J.-C. Janodet, and F. Tantini. Learning languages from bounded resources: The case of the DFA and the balls of strings. In *Proc. of the 9th International Colloquium in Grammatical Inference (ICGI'08)*, page ? (to appear). LNAI, 2008.

R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

C. L. Giles, S. Lawrence, and A.C. Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning Journal*, 44(1):161–183, 2001.

D. Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

A. Hagerer, H. Hungar, O. Niese, and B. Steffen. Model generation by moderated regular extrapolation. In *Proc. of the 5th International Conference on Fundamental Approaches to Software Engineering (FASE'02)*, pages 80–95. LNCS 2306, 2002.

M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, 1978.

M. J. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM Journal of Computing*, 22(4):807–837, 1993.

E. B. Kinber. On learning regular expressions and patterns via membership and correction queries. In *Proc. of the 9th International Colloquium in Grammatical Inference (ICGI'08)*, page ? (to appear). LNAI, 2008.

T. Kohonen. Median strings. *Pattern Recognition Letters*, 3:309–313, 1985.

K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *Proc. of the 4th International Colloquium in Grammatical Inference (ICGI'98)*, pages 1–12. LNAI 1433, 1998.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.

S. Lucas. Learning deterministic finite automata from noisy data competition, 2004. URL `http://cswww.essex.ac.uk/staff/sml/gecco/NoisyDFA.html`.

C. D. Martínez-Hinarejos, A. Juan, and F. Casacuberta. Use of median string for classification. In *Proc. of the 15th International Conference on Pattern Recognition (ICPR'00)*, volume 2, pages 2903–2906, 2000.

L. Mascheroni. Geometria del compasso. Pavia, 1797.

B. Melichar. Approximate string matching by finite automata. In *Proc. 6th International Conference on Computer Analysis of Images and Patterns (CAIP'95)*, pages 342–349. LNCS 970, 1995.

G. Mohr. Euclides danicus. Amsterdam, 1672.

G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

M.-F. Sagot and Y. Wakabayashi. Pattern inference under many Guises. In *Recent Advances in Algorithms and Combinatorics*, pages 245–287. Springer-Verlag, 2003.

J. Sakarovich. *Eléments de Théorie des Automates*. Vuibert, 2004.

A. Salomaa. On languages defined by numerical parameters. In *Formal Models, Languages and Applications*, volume 66 of *Machine Perception and Artificial Intelligence*, chapter 8. World Scientific Publishing Company, 2006.

K. U. Schulz and S. Mihov. Fast string correction with Levenshtein automata. *International Journal on Document Analysis and Recognition*, 5(1):67–85, 2002.

F. Tantini, C. de la Higuera, and J. C. Janodet. Identification in the limit of systematic-noisy languages. In *Proc. of the 8th International Colloquium in Grammatical Inference (ICGI'06)*, pages 19–31. LNCS 4201, 2006.

B. Trakhtenbrot and Y. Barzdin. *Finite Automata: Behavior and Synthesis*. North Holland Pub. Comp., Amsterdam, 1973.

E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, 1985.

R. Wagner and M. Fisher. The string-to-string correction problem. *Journal of the ACM*, 21:168–178, 1974.

Journal of Machine Learning Research 9 (2008) 1871-1874 Submitted 5/08; Published 8/08

LIBLINEAR: A Library for Large Linear Classification

Rong-En Fan B90098@CSIE.NTU.EDU.TW
Kai-Wei Chang B92084@CSIE.NTU.EDU.TW
Cho-Jui Hsieh B92085@CSIE.NTU.EDU.TW
Xiang-Rui Wang R95073@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Editor: Soeren Sonnenburg

Abstract

LIBLINEAR is an open source library for large-scale linear classification. It supports logistic regression and linear support vector machines. We provide easy-to-use command-line tools and library calls for users and developers. Comprehensive documents are available for both beginners and advanced users. Experiments demonstrate that LIBLINEAR is very efficient on large sparse data sets.

Keywords: large-scale linear classification, logistic regression, support vector machines, open source, machine learning

1. Introduction

Solving large-scale classification problems is crucial in many applications such as text classification. Linear classification has become one of the most promising learning techniques for large sparse data with a huge number of instances and features. We develop LIBLINEAR as an easy-to-use tool to deal with such data. It supports L2-regularized logistic regression (LR), L2-loss and L1-loss linear support vector machines (SVMs) (Boser et al., 1992). It inherits many features of the popular SVM library LIBSVM (Chang and Lin, 2001) such as simple usage, rich documentation, and open source license (the BSD license[1]). LIBLINEAR is very efficient for training large-scale problems. For example, it takes only several *seconds* to train a text classification problem from the Reuters Corpus Volume 1 (rcv1) that has more than 600,000 examples. For the same task, a general SVM solver such as LIBSVM would take several hours. Moreover, LIBLINEAR is competitive with or even faster than state of the art linear classifiers such as Pegasos (Shalev-Shwartz et al., 2007) and SVM$^{\text{perf}}$ (Joachims, 2006). The software is available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

This article is organized as follows. In Sections 2 and 3, we discuss the design and implementation of LIBLINEAR. We show the performance comparisons in Section 4. Closing remarks are in Section 5.

1. The New BSD license approved by the Open Source Initiative.

2. Large Linear Classification (Binary and Multi-class)

LIBLINEAR supports two popular binary linear classifiers: LR and linear SVM. Given a set of instance-label pairs $(\mathbf{x}_i, y_i), i = 1, \ldots, l,\ \mathbf{x}_i \in R^n,\ y_i \in \{-1, +1\}$, both methods solve the following unconstrained optimization problem with different loss functions $\xi(\mathbf{w}; \mathbf{x}_i, y_i)$:

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi(\mathbf{w}; \mathbf{x}_i, y_i), \tag{1}$$

where $C > 0$ is a penalty parameter. For SVM, the two common loss functions are $\max(1 - y_i\mathbf{w}^T\mathbf{x}_i, 0)$ and $\max(1 - y_i\mathbf{w}^T\mathbf{x}_i, 0)^2$. The former is referred to as L1-SVM, while the latter is L2-SVM. For LR, the loss function is $\log(1 + e^{-y_i\mathbf{w}^T\mathbf{x}_i})$, which is derived from a probabilistic model. In some cases, the discriminant function of the classifier includes a bias term, b. LIBLINEAR handles this term by augmenting the vector $\mathbf{w}$ and each instance $\mathbf{x}_i$ with an additional dimension: $\mathbf{w}^T \leftarrow [\mathbf{w}^T, b], \mathbf{x}_i^T \leftarrow [\mathbf{x}_i^T, B]$, where B is a constant specified by the user. The approach for L1-SVM and L2-SVM is a coordinate descent method (Hsieh et al., 2008). For LR and also L2-SVM, LIBLINEAR implements a trust region Newton method (Lin et al., 2008). The Appendix of our SVM guide.[2] discusses when to use which method. In the testing phase, we predict a data point $\mathbf{x}$ as positive if $\mathbf{w}^T\mathbf{x} > 0$, and negative otherwise. For multi-class problems, we implement the one-vs-the-rest strategy and a method by Crammer and Singer. Details are in Keerthi et al. (2008).

3. The Software Package

The LIBLINEAR package includes a library and command-line tools for the learning task. The design is highly inspired by the LIBSVM package. They share similar usage as well as application program interfaces (APIs), so users/developers can easily use both packages. However, their models after training are quite different (in particular, LIBLINEAR stores $\mathbf{w}$ in the model, but LIBSVM does not.). Because of such differences, we decide not to combine these two packages together. In this section, we show various aspects of LIBLINEAR.

3.1 Practical Usage

To illustrate the training and testing procedure, we take the data set news20,[3] which has more than one million features. We use the default classifier L2-SVM.

```
$ train news20.binary.tr
[output skipped]
$ predict news20.binary.t news20.binary.tr.model prediction
Accuracy = 96.575% (3863/4000)
```

The whole procedure (training and testing) takes less than 15 seconds on a modern computer. The training time without including disk I/O is less than one second. Beyond this simple way of running LIBLINEAR, several parameters are available for advanced use. For example, one may specify a parameter to obtain probability outputs for logistic regression. Details can be found in the README file.

2. The guide can be found at http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.
3. This is the news20.binary set from http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets. We use a 80/20 split for training and testing.

3.2 Documentation

The LIBLINEAR package comes with plenty of documentation. The `README` file describes the installation process, command-line usage, and the library calls. Users can read the "Quick Start" section, and begin within a few minutes. For developers who use LIBLINEAR in their software, the API document is in the "Library Usage" section. All the interface functions and related data structures are explained in detail. Programs `train.c` and `predict.c` are good examples of using LIBLINEAR APIs. If the `README` file does not give the information users want, they can check the online FAQ page.[4] In addition to software documentation, theoretical properties of the algorithms and comparisons to other methods are in Lin et al. (2008) and Hsieh et al. (2008). The authors are also willing to answer any further questions.

3.3 Design

The main design principle is to keep the whole package as simple as possible while making the source codes easy to read and maintain. Files in LIBLINEAR can be separated into source files, pre-built binaries, documentation, and language bindings. All source codes follow the C/C++ standard, and there is no dependency on external libraries. Therefore, LIBLINEAR can run on almost every platform. We provide a simple `Makefile` to compile the package from source codes. For Windows users, we include pre-built binaries.

Library calls are implemented in the file `linear.cpp`. The `train()` function trains a classifier on the given data and the `predict()` function predicts a given instance. To handle multi-class problems via the one-vs-the-rest strategy, `train()` conducts several binary classifications, each of which is by calling the `train_one()` function. `train_one()` then invokes the solver of users' choice. Implementations follow the algorithm descriptions in Lin et al. (2008) and Hsieh et al. (2008). As LIBLINEAR is written in a modular way, a new solver can be easily plugged in. This makes LIBLINEAR not only a machine learning tool but also an experimental platform.

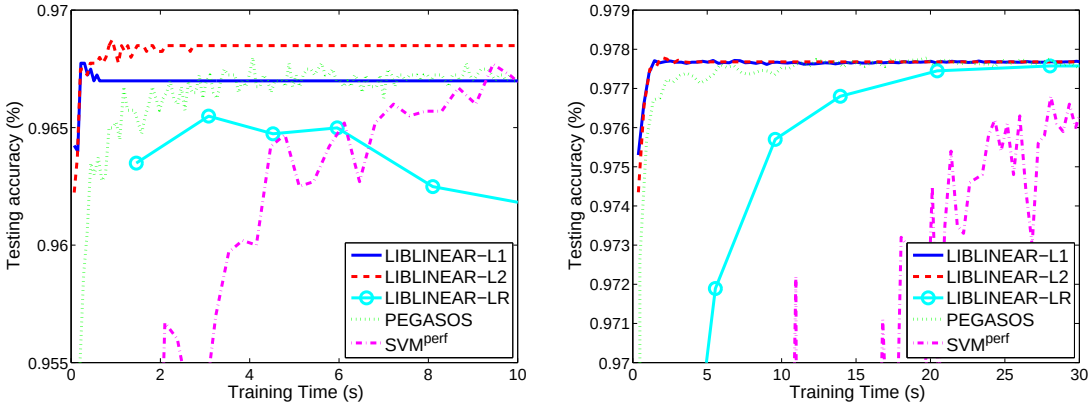
Making extensions of LIBLINEAR to languages other than C/C++ is easy. Following the same setting of the LIBSVM MATLAB/Octave interface, we have a MATLAB/Octave extension available within the package. Many tools designed for LIBSVM can be reused with small modifications. Some examples are the parameter selection tool and the data format checking tool.

4. Comparison

Due to space limitation, we skip here the full details, which are in Lin et al. (2008) and Hsieh et al. (2008). We only demonstrate that LIBLINEAR quickly reaches the testing accuracy corresponding to the optimal solution of (1). We conduct five-fold cross validation to select the best parameter C for each learning method (L1-SVM, L2-SVM, LR); then we train on the whole training set and predict the testing set. Figure 1 shows the comparison between LIBLINEAR and two state of the art L1-SVM solvers: Pegasos (Shalev-Shwartz et al., 2007) and SVM$^{\text{perf}}$ (Joachims, 2006). Clearly, LIBLINEAR is efficient.

To make the comparison reproducible, codes used for experiments in Lin et al. (2008) and Hsieh et al. (2008) are available at the LIBLINEAR web page.

4. FAQ can be found at `http://www.csie.ntu.edu.tw/~cjlin/liblinear/FAQ.html`.



(a) news20, l: 19,996, n: 1,355,191, #nz: 9,097,916 (b) rcv1, l: 677,399, n: 47,236, #nz: 156,436,656

Figure 1: Testing accuracy versus training time (in seconds). Data statistics are listed after the data set name. l: number of instances, n: number of features, #nz: number of nonzero feature values. We split each set to 4/5 training and 1/5 testing.

5. Conclusions

LIBLINEAR is a simple and easy-to-use open source package for large linear classification. Experiments and analysis in Lin et al. (2008), Hsieh et al. (2008) and Keerthi et al. (2008) conclude that solvers in LIBLINEAR perform well in practice and have good theoretical properties. LIBLINEAR is still being improved by new research results and suggestions from users. The ultimate goal is to make easy learning with huge data possible.

References

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *COLT*, 1992.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML*, 2008.

T. Joachims. Training linear SVMs in linear time. In *ACM KDD*, 2006.

S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A sequential dual method for large scale multi-class linear SVMs. In *ACM KDD*, 2008.

C.-J. Lin, R. C. Weng, and S. S. Keerthi. Trust region Newton method for large-scale logistic regression. *JMLR*, 9:627–650, 2008.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: primal estimated sub-gradient solver for SVM. In *ICML*, 2007.

Journal of Machine Learning Research 9 (2008) 1875-1908 Submitted 4/07; Revised 4/08; Published 8/08

On Relevant Dimensions in Kernel Feature Spaces

Mikio L. Braun MIKIO@CS.TU-BERLIN.DE
Technische Universität Berlin
Franklinstr. 28/29, FR 6-9
10587 Berlin, Germany

Joachim M. Buhmann JBUHMANN@INF.ETHZ.CH
Institute of Computational Science
ETH Zurich, Universitätstrasse 6
CH-8092 Zürich, Switzerland

Klaus-Robert Müller[*] KRM@CS.TU-BERLIN.DE
Technische Universität Berlin
Franklinstr. 28/29, FR 6-9
10587 Berlin, Germany

Editor: Peter Bartlett

Abstract

We show that the relevant information of a supervised learning problem is contained up to negligible error in a finite number of leading kernel PCA components if the kernel matches the underlying learning problem in the sense that it can asymptotically represent the function to be learned and is sufficiently smooth. Thus, kernels do not only transform data sets such that good generalization can be achieved using only linear discriminant functions, but this transformation is also performed in a manner which makes economical use of feature space dimensions. In the best case, kernels provide efficient implicit representations of the data for supervised learning problems. Practically, we propose an algorithm which enables us to recover the number of leading kernel PCA components relevant for good classification. Our algorithm can therefore be applied (1) to analyze the interplay of data set and kernel in a geometric fashion, (2) to aid in model selection, and (3) to denoise in feature space in order to yield better classification results.

Keywords: kernel methods, feature space, dimension reduction, effective dimensionality

1. Introduction

Kernel machines implicitly map the data into a high-dimensional feature space in a non-linear fashion using a kernel function. This mapping is often referred to as an *empirical kernel map* (Schölkopf et al., 1999; Vapnik, 1998; Müller et al., 2001; Schölkopf and Smola, 2002). By virtue of the empirical kernel map, the data is ideally transformed in a way such that a linear discriminative function can separate the classes with low generalization error by a canonical hyperplane with large margin. Such large margin hyperplanes provide an appropriate mechanism of capacity control and thus "protect" against the high dimensionality of the feature space.

However, this picture is incomplete as it does not explain why the typical variants of capacity control cooperate well with the induced feature map. This paper adds a novel aspect as the key idea

[*]. Also at Fraunhofer FIRST.IDA, Kekuléstr. 7, 12489 Berlin, Germany.

to this picture. We show theoretically that if the learning problem matches the kernel well, the relevant information of a supervised learning data set is always contained in the subspace spanned by a finite and typically small number of leading kernel PCA components (principal component analysis in the feature space induced by the kernel, see below and Section 2), up to negligible error. This result is based on recent approximation bounds for the eigenvectors of the kernel matrix which show that if a function can be reconstructed using only a few kernel PCA components asymptotically, then the same already holds in a finite sample setting, even for small sample sizes.

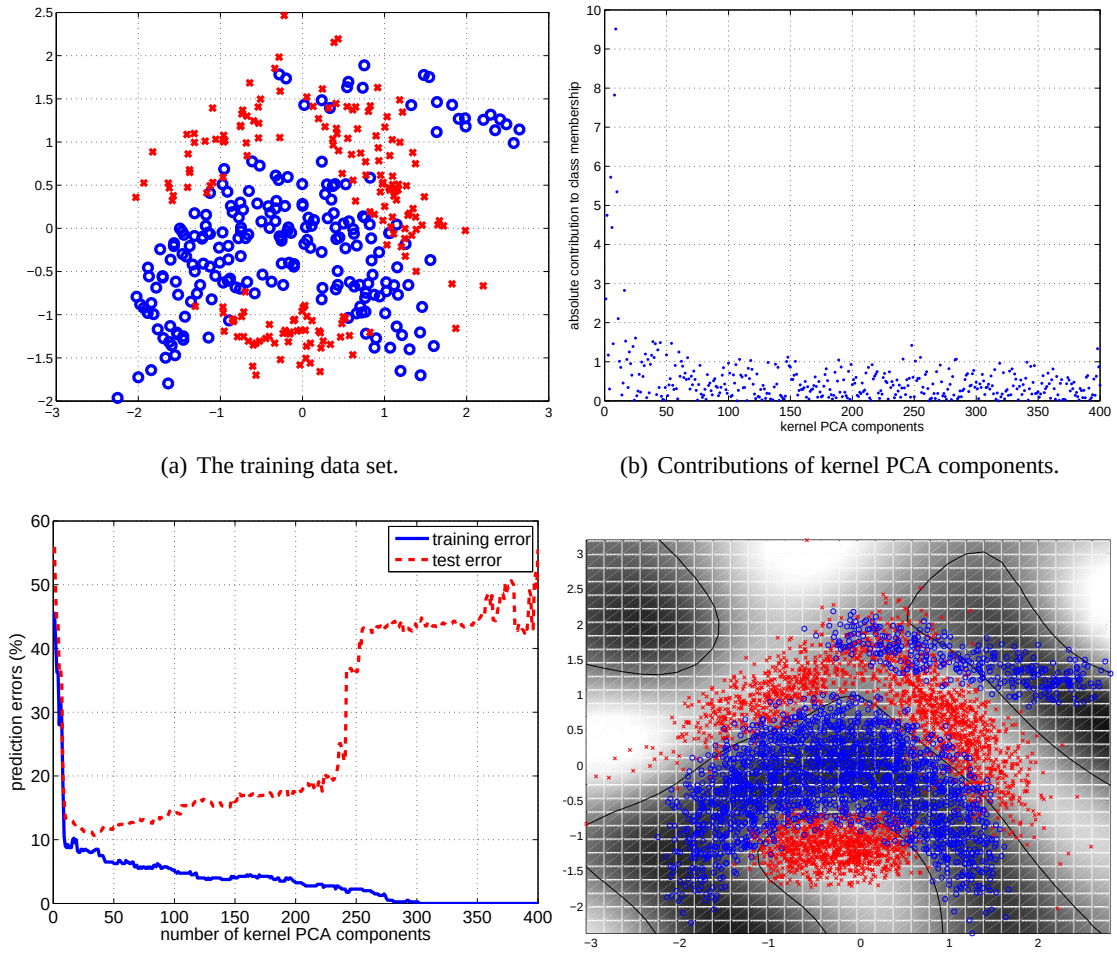
Consequently, the use of a kernel function not only greatly increases the expressive power of linear methods by non-linearly transforming the data, but it does so ensuring that the high dimensionality of the feature space does not become overwhelming: the relevant information for learning stays confined within a comparably *low*-dimensional subspace. This finding underlines the efficient use of data that is made by kernel machines if the kernel works well for the learning problem. A smart choice of kernel permits to make better use of the available data at a favorable "number of data points per effective dimension"-ratio, even for infinite-dimensional feature spaces. The kernel induces an efficient representation of the data in feature space such that even unregularized methods like linear least squares regression are able to perform well on the reduced feature space.

Let us consider an example. Figure 1(a) shows a two-dimensional classification problem (the *banana* data set from Rätsch et al., 2001). We can visualize the contributions of the individual kernel PCA components[1] to the class membership by plotting the absolute values of scalar products between the labels and the kernel PCA components. Figure 1(b) shows the resulting contributions sorted by decreasing principal value (variance along principal direction). We can observe that the contributions are concentrated in the leading kernel PCA directions, but a large fraction of the information is contained in the later components as well.

Note, however, that the class membership information in the data set also contains a certain amount of noise. Therefore, Figure 1(b) actually shows a mixture of relevant information and noise. We need to devise a different procedure for assessing the amount of task-relevant information in certain kernel PCA components. This can be accomplished by incorporating a second data set from the same source for testing. One first projects onto the subspace spanned by a number of leading kernel PCA components, trains a linear classifier (for example, by least squares regression) and then measures the prediction error on the test set. The test error is large either if the considered subspace did not capture all of the relevant information, or if it already contained too much noise leading to overfitting. If the minimal test error is on par with a state-of-the-art method independently trained using the same kernel then the subspace has successfully captured all of the relevant information.

If we apply this procedure to our data set, we obtain training and test errors as shown in Figure 1(c). By definition, the training error decreases as more and more dimensions are used. However, after decreasing quickly initially, the test error eventually starts to increase again. The minimal test error also coincides with the actually achievable test error using, for example, support vector machines. Therefore, we see that the later components only contain noise, and the relevant information is contained in the leading kernel PCA components. In this paper, our goal is to understand

1. Recall that kernel PCA (Schölkopf et al., 1998) amounts to implicitly performing PCA in the feature space. Roughly, instead of the covariance matrix, one considers the eigenvalues and eigenvectors of the kernel matrix, which is built from all pairwise evaluations of the kernel matrix on the inputs. Principal values (variances) are still given by the eigenvalues of the kernel matrix, but principal directions (which would be potentially infinite-dimensional vectors) are replaced by principal components, which are scalar products with the principal directions. Also see Section 2.



(a) The training data set.

(b) Contributions of kernel PCA components.

(c) Training and test errors using only leading kernel PCA components.

(d) The solution on the test data set.

Figure 1: A more complex example (resample 1 of the "banana" data set, see Section A). This time, the information is not contained in a single component. Nevertheless, the test error of a hyperplane learned using only the first d components has a clear minimum at $d = 34$ at optimal error rate (cf. Table 3), showing that the relevant information is contained in the leading 34 directions.

more thoroughly why and when this effect occurs, and to estimate the dimensionality of a concrete data set given a kernel.

Our claim—that the relevant information about a learning problem is contained in the space spanned by the leading kernel PCA components—is similar to the idea that the information about the learning problem is contained in the kernel PCA components with the largest contributions. However, our results show that the magnitude of the contribution of a kernel PCA component to the label information is only partially indicative of the relevance of that component. Instead, we show that the leading kernel PCA components (sorted by corresponding principal value) contain

the relevant information. Components which contain only little variance will therefore not contain relevant information. If such a component manages to contribute much to the label information, it will only reflect noise.

What practical implications follow from these results? We explore several possibilities of using these ideas to assess the suitability of a kernel or a family of kernels to a specific data set. The main idea is that the observed dimensionality of the data set in feature space is characteristic for the relation between a data set and a kernel. Roughly speaking, the relevant dimensionality of the data set corresponds to the complexity of the learning problem when viewed through the "lens" of the kernel function. Using the estimated dimensionality, one can project the labels onto the corresponding subspace and obtain a noise free version of the labels. By comparing the denoised labels to the original labels, one can estimate of the amount of noise contained in the labels. One therefore obtains a more detailed measure of the fit between the kernel and the data set as compared to, for example, the cross-validation error alone. This allows us to take a closer look at data sets on which the achieved error is quite large. In such cases, we are able to distinguish whether the data set is highly complex and the amount of data is insufficient, or the amount of intrinsic noise is very large. This is practically relevant as one has to deal with both these cases quite differently, either by providing more data, or by thinking about means to obtain less noisy or ambiguous features.

We summarize the main contributions of this paper: (1) We provide theoretical bounds showing that the relevant information (defined in Section 2) is actually contained in the leading projected kernel principal components under appropriate conditions. (2) We propose an algorithm which estimates the relevant dimensionality and related estimates of the data set and permits to analyze the appropriateness of a kernel for the data set, and thus to perform model selection among different kernels. (3) We validate the accuracy of the estimates experimentally by showing that non-regularized methods perform on the reduced feature space on par with state-of-the-art kernel methods. We analyze some well-known benchmark data sets in Section 5. Note that we do not claim to obtain better performance within our framework when compared to, for example, cross-validation techniques. Rather, we are on par. Our contribution is to foster an understanding about a data set and to gain better insights of whether a mediocre classification result is due to intrinsic high dimensionality of the data (and consequently insufficient number of examples), or an overwhelming noise level.

2. Preliminaries

Let us start to formalize the ideas introduced so far. As usual, we consider a data set (X_1,Y_1), $\ldots,(X_n,Y_n)$ where the inputs X lie in some space $\mathcal{X}$ and the outputs Y to be predicted are in $\mathcal{Y} = \{\pm 1\}$ for classification or $\mathcal{Y} = \mathbb{R}$ for regression. We often refer to the outputs Y_i as the "labels" irrespective of whether we are considering a classification or regression task. We assume that the (X_i,Y_i) are drawn i.i.d. from some probability measure $\mathrm{P}_{\mathcal{X}\times\mathcal{Y}}$. In kernel methods, the data is nonlinearly mapped into some feature space $\mathcal{F}$ via the feature map Φ. Scalar products in $\mathcal{F}$ can be computed by the kernel k in closed form: $\langle \Phi(x), \Phi(x')\rangle = k(x,x')$. Summarizing all the pairwise scalar products results in the (normalized) kernel matrix $\mathbf{K}$ with entries $k(X_i,X_j)/n$.

In the discussion below, we study the relationship between the *label vector* $Y = (Y_1,\ldots,Y_n)$ and the kernel PCA components which are introduced next. Kernel PCA (Schölkopf et al., 1998) is a kernelized version of PCA. Since the dimensionality of the feature space might be too large to deal

Symbol	Meaning
n	number of training examples
$X_i \in \mathcal{X}$	input examples
$Y_i \in \mathcal{Y}$	output labels
$k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	kernel function
$\mathbf{K} = (k(X_i, X_j))/n$	(normalized) kernel matrix
$Y = (Y_1, \ldots, Y_n)$	label vector
$\Phi\colon \mathcal{X} \to \mathcal{F}$	feature map
$l_m \in \mathbb{R}_{\geq 0}$	mth kernel PCA value (in descending order), mth eigenvalue of kernel matrix $\mathbf{K}$
$v_m \in \mathcal{F}$	mth kernel PCA direction
$f_m(x) = \langle \Phi(x), v_m \rangle$	mth kernel PCA component
$u_m = (f_m(X_1), \ldots, f_m(X_n))$	mth kernel PCA component evaluated on $X_1, \ldots, X_n$, mth eigenvector of kernel matrix $\mathbf{K}$
$\pi_d(Y) = \sum_{i=1}^{d} u_i u_i Y$	projection onto first d kernel PCA components
$G = (\mathrm{E}(Y_1 \mid X_1), \ldots, \mathrm{E}(Y_n \mid X_n))$	relevant information vector
$z_i = u_i^\top G$	contribution of ith eigenvector to relevant information
$g(x) = \mathrm{E}(Y \mid X = x)$	relevant information function
$\mathcal{L}^2(\mathcal{X}, \mathrm{P}_\mathcal{X})$	set of all square integrable functions with respect to $\mathrm{P}_\mathcal{X}$
$T_k f(s) = \int_\mathcal{X} k(s,t) f(t) \mathrm{P}_\mathcal{X}(dt)$	integral operator associated with k
$\lambda_i \in \mathbb{R}_{\geq 0}$	ith eigenvalue of T_k
$\psi_i \in \mathcal{L}^2(\mathcal{X}, \mathrm{P}_\mathcal{X})$	ith eigenfunction of T_k
$\zeta_i = \langle \psi_i, g \rangle$	contribution of ith eigenfunction to relevant information
$\hat{d}$	estimated relevant dimension
$\mathrm{cv}_{\mathrm{loo}}$	leave-one-out cross-validation error
$\hat{G}$	estimated relevant information vector
$\mathbf{S} = \sum_{i=1}^{d} u_i u_i^\top$	"hat"-matrix
$\hat{\mathrm{err}}$	estimated noise-level

Table 1: Overview of notation used in this paper.

with the vectors directly, the principal directions are represented using the points X_i of the data set:

$$v_m = \sum_{i=1}^{n} \alpha_i \Phi(X_i),$$

where $\alpha_i = [u_m]_i / l_m$, $[u_m]_i$ is the ith component of the mth eigenvector of the kernel matrix $\mathbf{K}$, and l_m the corresponding eigenvalue.[2] Still, v_m can usually not be computed explicitly such that one instead works with *kernel PCA components*

$$f_m(x) = \langle \Phi(x), v_m \rangle.$$

We are interested in the relation between f_m and a label vector Y. As we have seen in the introduction, it seems that only a finite number of leading kernel PCA components are necessary to represent the relevant information about the learning problem up to a small error.

2. As usual, we assume that l_m and u_m have been sorted such that $l_1 \geq \ldots \geq l_n$.

Therefore, we would like to compare f_m with the values $Y_1,\ldots,Y_n$ at the points $X_1,\ldots,X_n$. The following easy lemma summarizes the relationship between the sample vector of f_m and Y.

Lemma 1 *The mth kernel PCA component f_m evaluated on the X_is is equal to the mth eigenvector of the kernel matrix* $\mathbf{K}$*: $(f_m(X_1),\ldots,f_m(X_n)) = u_m$. Consequently, the sample vectors are orthogonal, and the projection of a vector $Y \in \mathbb{R}^n$ onto the leading d kernel PCA components is given by $\pi_d(Y) = \sum_{m=1}^{d} u_m u_m^\top Y$.*

Proof The mth kernel PCA component for a point X_j in the training set is

$$f_m(X_j) = \langle \Phi(X_j), v_m \rangle = \frac{1}{l_m} \sum_{i=1}^{n} \langle \Phi(X_j), \Phi(X_i) \rangle [u_m]_i = \frac{1}{l_m} \sum_{i=1}^{n} k(X_j, X_i)[u_m]_i.$$

The sum computes the jth component of $\mathbf{K}u_m$, and $\mathbf{K}u_m = l_m u_m$, because u_m is an eigenvector of $\mathbf{K}$. Therefore

$$f_m(X_j) = \frac{1}{l_m}[l_m u_m]_j = [u_m]_j.$$

Since $\mathbf{K}$ is a symmetric matrix, its eigenvectors u_m are orthonormal, and the projection of Y onto the space spanned by the first d kernel PCA components is given by $\sum_{m=1}^{d} u_m u_m^\top Y$. ■

Since the kernel PCA components are orthogonal, the coefficients of a vector $Y \in \mathbb{R}^n$ with respect to the basis $u_1,\ldots,u_n$ is easily computed by forming the scalar products. We call the coefficients

$$z_m = u_m^\top Y \tag{1}$$

of Y w.r.t. the basis formed from the kernel PCA components the *kernel PCA coefficients*. They are the central object of our discussion.

The projection of Y to a kernel PCA component can be thought of as the least squares regression of Y using only the direction along the kernel PCA component in feature space.

Using the kernel PCA coefficients, we can extend the projected labels to new points via

$$\hat{Y}(x) = \sum_{m=1}^{d} z_m f_m(x),$$

which amounts to the prediction of least squares regression on the reduced feature space.

3. The Label Vector and Kernel PCA Components

In the introduction, we have discussed an example which suggests that a small number of leading kernel PCA components might suffice to capture the relevant information about the output variable. It is clear that we cannot expect this behavior for all possible data sets and kernels. It seems plausible though, that under certain conditions, the distribution of the data and the kernel fit together well. Then we can expect to observe this behavior with high probability for a random sample from this distribution through some form of concentration or convergence property.

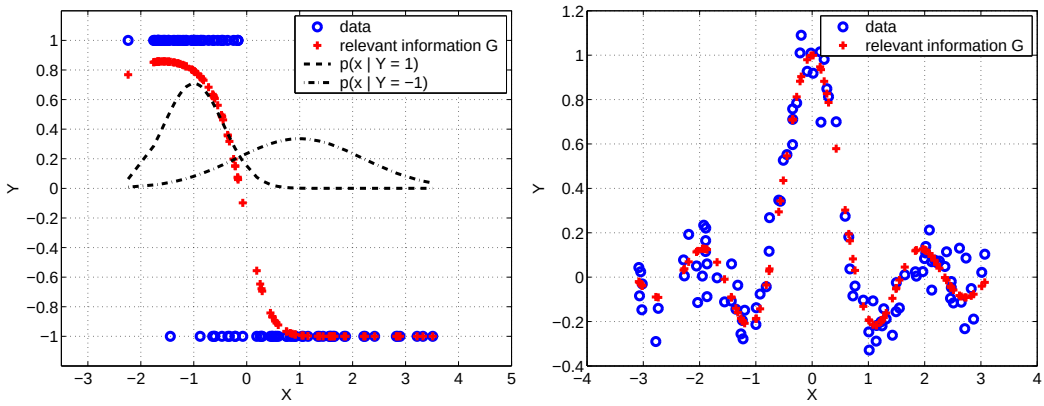


Figure 2: Relevant information vectors visualized for the classification and the regression case. In the (two-class) classification case (left) it encodes the posterior probability (scaled between -1 and 1), in the regression case it is the sample vector of the function to be learned.

3.1 Decomposing the Label Vector Information

We start the discussion by defining formally what the relevant information contained in the labels is. Given a label vector Y, we define the relevant information vector as the vector of the expected labels:

$$G = (\mathrm{E}(Y_1|X_1), \ldots, \mathrm{E}(Y_n|X_n)).$$

Intuitively speaking, G is a noise-free version of Y. This vector contains all the relevant information about the outputs Y of the learning problem: For regression, G amounts to the values of the true function. For the case of two-class classification, the vector G contains all the information about the optimal decision boundary. Since $\mathrm{E}(Y|X) = \mathrm{P}(Y = 1|X) - \mathrm{P}(Y = -1|X)$, the sign of G contains the relevant information on the true class membership by telling us which class is more probable (see Figure 2 for examples). Thus, using this denoised label information, the learning problem becomes much easier as the denoised labels already contain the Bayes optimal prediction at that point.[3]

Using G we obtain a very useful additive decomposition of the labels into "signal" and "noise":

$$Y = G + N.$$

In this setting, we are now interested in showing that G is contained in the leading kernel PCA components, such that projecting G onto the leading kernel PCA components leads to only negligible error. In the following, we treat the signal and noise part of Y separately. This is possible because the projection π_d is a linear operation such that $\pi_d(Y) = \pi_d(G+N) = \pi_d(G) + \pi_d(N)$.

3. Also note that the capacity control typically employed in kernel methods amounts to some form of regularization, or "implicit denoising" (Smola et al., 1998). Therefore, we do not expect that the results using G are generally better than with the original labels. However, as we will see below, *unregularized* methods perform on par with kernel methods with capacity control using the estimated relevant information vector G.

3.2 The Relevant Information Vector

We first treat the relevant information vector G. The location of G with respect to the kernel PCA components is characterized by scalar products with the eigenvectors of the kernel matrix. We start by discussing this relationship in an asymptotic setting and then transfer the results back to the finite sample setting using convergence results for the spectral properties of the kernel matrix

Using the kernel function k, we define the integral operator

$$T_k f(s) = \int_{\mathcal{X}} k(s,t) f(t) \mathrm{P}_{\mathcal{X}}(dt),$$

where $\mathrm{P}_{\mathcal{X}}$ is the marginal distribution which generates the inputs X_i. It is well known that the linear operator

$$\tilde{T}_k f(s) = \frac{1}{n} \sum_{i=1}^{n} k(s, X_i) f(X_i)$$

represented by the kernel matrix approximates T_k as the number of points tend to infinity (see, for example, von Luxburg, 2004). While this follows easily for a fixed f and s, making the argument theoretically exact for operators (this means uniform over all functions) is not trivial.

As a consequence, the eigenvalues and eigenvectors of $\tilde{T}_k$, which are equal to those of the kernel matrix, converge to those of T_k (see Koltchinskii and Giné, 2000; Koltchinskii, 1998). In particular, scalar products of sample functions and eigenvectors of $\mathbf{K}$ converge to scalar products with eigenfunctions of T_k. The asymptotic counterpart of the relevant information vector G is the function

$$g(x) = \mathrm{E}(Y|X = x).$$

These correspondences are summarized in Figure 3. In summary, we can think of $z_i = u_i^\top G$ (properly scaled) as an approximation to $\zeta_i = \langle \psi_i, g \rangle$.

finite sample setting
$[\mathbf{K}x]_i = \frac{1}{n} \sum_{j=1}^{n} k(X_i, X_j) x_j$
l_i eigenvalue of $\mathbf{K}$
u_i eigenvector of $\mathbf{K}$
$G = \big(\mathrm{E}(Y_1\|X_1), \ldots \mathrm{E}(Y_n\|X_n)\big)$
$z_i = u_i^\top G$

$\rightsquigarrow$

asymptotic setting
$T_k f(s) = \int_{\mathcal{X}} k(s,t) f(t) \mathrm{P}_{\mathcal{X}}(dt)$
λ_i eigenvalue of T_k
ψ_i eigenfunction of T_k
$g(x) = \mathrm{E}(Y\|X = x)$
$\zeta_i = \langle \psi_i, g \rangle$

Figure 3: Transition from the finite sample size and asymptotic setting.

In the asymptotic setting, it is now fairly easy to specify conditions such that g is contained in the subspace spanned by a finite number of leading eigenfunctions ψ_i. Since it is unrealistic that g is exactly contained in a finite dimensional subspace, we relax that requirement and instead only require that ζ_i decays to zero at the same rate as the eigenvalues of T_k.

The decay rate of the eigenvalues depends on the interplay between the kernel and the distribution $\mathrm{P}_{\mathcal{X}}$. However, expressing this connection in closed form is in general not possible. As a rule of thumb, the eigenvalues decay quickly when the kernel is smooth at the scale of the data. Since one usually uses smooth kernels to prevent from overfitting, the eigenvalues typically decay rather quickly. As we will see, most of the information about g is then contained in a few kernel PCA components.

A natural assumption is that the learning problem can be asymptotically represented by the given kernel function k. By this we mean that there exists some function $h \in \mathcal{L}^2(\mathcal{X}, \mathrm{P}_\mathcal{X})$ such that $g = T_k h$. Using the spectral decomposition of T_k, this implies

$$g = T_k h = \sum_{i=1}^{\infty} \lambda_i \langle \psi_i, h \rangle \psi_i. \tag{2}$$

Since the sequence of $\alpha_i = \langle \psi_i, h \rangle$ is square summable, it follows that

$$\zeta_i = \langle \psi_i, g \rangle = \lambda_i \alpha_i = O(\lambda_i).$$

Intuitively speaking, (2) translates to asymptotic representability of the learning problem: As $n \to \infty$, it becomes possible to represent the optimal labels using the kernel function k.

Furthermore, we assume that k is bounded. This technical requirement is mainly necessary to ensure that g is also bounded. The requirement holds for common radial basis function kernels like the Gaussian kernel, and also if the underlying space $\mathcal{X}$ is compact and the kernel is continuous.

Note that the requirement that g lies in the range of T_k is essential. If this is not the case, we cannot expect that the scalar products decay at a given rate. Also note that it is in fact possible to break this condition. For example, if k is continuous, every non-continuous function does not lie in the range of T_k.

The question is now whether the same behavior can be expected for a finite data set. This question is not trivial, because eigenvector stability is known to be linked to the gap between the corresponding eigenvalues, which is fairly small for small eigenvalues (see, for example, Zwald and Blanchard, 2006).

The main theoretical result of this paper (Theorem 1 in the Appendix) provides a bound of the form

$$\frac{1}{n} |u_i^\top G| \leq l_i C + E$$

which expresses an essential equivalence between the finite sample setting and the asymptotic setting with two modifications: The decay rate $O(\lambda_i)$ of the scalar products $\langle \psi_i, g \rangle$ holds for the finite sample up to a (small) additive error E with λ_i replaced by its finite sample approximation l_i.

The technical details of this theorem and the proof are deferred to the appendix. Let us discuss how the absolute term occurs in the bound and why it can be expected to be small. An exact scaling bound (without additive term E) can only be derived (at least following the approach taken in this paper) for the case where the kernel function is degenerate, that is, T_k has only finitely many non-zero eigenvalues. The same finiteness restriction also holds for the expansion of g in terms of the eigenfunctions of T_k. The proof thus contains a truncation step of general kernels and general functions g, leading to a scaling bound on the scalar product and an additive term arising from the truncation. However, as the name suggests, the truncation error E can be made arbitrarily small by considering approximations with many non-zero eigenvalues. At the same time, considering such kernels with more terms in the expansion leads to a larger constant C in the actual scaling part. Thus, both terms have to be balanced by the order of truncation, which permits to control the additive term well practically.

Note that the problem considered here is significantly different from the problem studying the performance of kernel PCA itself (see, for example, Blanchard et al., 2007; Shawe-Taylor et al., 2005; Mika, 2002). There, only the projection error using the Xs is studied. Here, we are specifically interested in the relationship between the Ys and the Xs.

In view of our original concern, the bound shows that the relevant information vector G (as introduced in Section 2) is contained in a number of leading kernel PCA components up to a negligible error. The number of dimensions depends on the asymptotic coefficients α_i and the decay rate of the asymptotic eigenvalues of k. Since this rate is related to the smoothness of the kernel function, the dimension is small for smooth kernels whose leading eigenfunctions ψ_i permit good approximation of g.

3.3 The Noise

To study the relationship between the noise and the eigenvectors of the kernel matrix, no asymptotic arguments are necessary. The key insight is that the eigenvectors are independent of the noise in the labels, such that the noise vector N is typically evenly distributed over all coefficients $u_i^\top N$: Let $\mathbf{U}$ be the matrix whose ith column is equal to u_i. The coefficients of N with respect to the eigenbasis of $\mathbf{K}$ are then given by $\mathbf{U}^\top N$. Note that since $\mathbf{U}$ is orthogonal, multiplication by its transpose amounts to a (random) rotation. In particular, this rotation is independent of the noise N as the u_i depend on the Xs only. Now if the noise has a spherical distribution, for example, N is normally distributed with covariance matrix $\sigma_\varepsilon^2\mathbf{I}$, it follows that $\mathbf{U}^\top N \sim \mathcal{N}(0,\sigma_\varepsilon^2\mathbf{I})$. For heteroscedastic noise in a regression setting, or for classification, this simple analysis is not sufficient. In that case, the individual $u_i^\top N$ are no longer uncorrelated. However, because of the independence of the N_i, the variance of $u_i^\top N$ is upper bounded by

$$\operatorname{Var}(u_i^\top N) = \sum_{j=1}^{n} u_{i,j}^2 \operatorname{Var}(N_j) \leq \max_{1\leq j\leq n} \operatorname{Var}(N_j)$$

since $\sum_{j=1}^n u_{i,j}^2 = \|u_i\|^2 = 1$. Therefore, the variance of the $u_i^\top N$ is not concentrated in any single coefficient as the total variance does not increase by rotating the basis and the individual variances are bounded by the maximum individual variance before the rotation.

The practical relevance of these observations is that the relevant information and noise part have radically different properties with respect to the kernel PCA components, allowing us to practically estimate the number of relevant dimension for a given kernel and data set. In the next section, we will propose two different algorithms for this task.

4. Relevant Dimension Estimation and Related Estimates

We have seen that the number of leading kernel PCA components necessary to capture the relevant information about the labels of a finite size data set is bounded under the mild assumptions that the learning problem can be represented asymptotically and the kernel is smooth such that the eigenvalues of the kernel matrix decay quickly. The actual number of necessary dimensions depends on the interplay between kernel and learning data set, giving insights into the suitability of the kernel. For example, a kernel might fail to provide an efficient representation of the learning problem, leading to an embedding requiring many kernel PCA components to capture the information on Y. Or, even worse, a kernel might completely fail to model some part of the learning problem, such that a part of the information appears to be just noise. Therefore, in order to make practical use of the presented insights, we need to devise a method to estimate the number of relevant kernel PCA components for a given concrete data set and choice of kernel.

In this section we propose methods for estimating the actual dimensionality of a data set, and two related estimators. Based on the dimensionality estimate, one can denoise the labels by projecting

onto the respective subspace and obtain an estimate for the relevant information vector G. By comparing the denoised labels with the original labels, one can then estimate the overall noise level of the data source. Based on these estimates, we discuss how to use the dimensionality estimate for model-selection and to further analyze data sets which so far show inferior performance. Figure 4 summarizes the information flow for the different estimates.

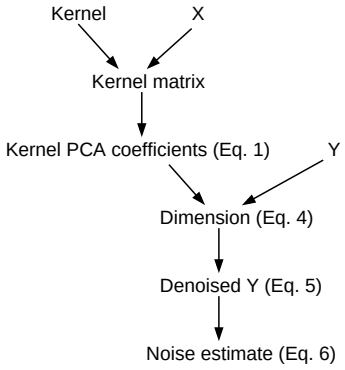


Figure 4: Information flow for the estimates.

4.1 Relevant Dimension Estimation (RDE)

The most basic estimate is the number of relevant kernel PCA components. We also call this number simply the *relevant dimension* or the *dimensionality* (also see the discussion in Section 6.3). Recall that we have decomposed the labels into $Y = G + N$, with $G_i = \mathrm{E}(Y_i|X_i)$ (see Section 3.1). This decomposition carries over to the kernel PCA coefficients $z_i = u_i^\top Y = u_i^\top G + u_i^\top N$. We want to estimate $\hat{d}$ such that $|u_i^\top G|$ is negligible for $i > \hat{d}$.

We propose two algorithms for solving this relevant dimension estimation (RDE) task which are based on different approaches to the problem but lead to comparable performance. The first algorithm fits a parametric model to the kernel PCA coefficients, while the second one is based on leave-one-out cross-validation.

4.1.1 RDE BY FITTING A TWO-COMPONENT MODEL (TCM)

The first algorithm works only on the coefficients $z_i = u_i^\top Y$. Recall that $\mathbf{U}$ is the matrix whose columns are the eigenvectors of the kernel matrix u_i such that $z = \mathbf{U}^\top Y = \mathbf{U}^\top G + \mathbf{U}^\top N = \tilde{G} + \tilde{N}$. In Section 3, we have seen that both parts have significantly different structure. From Theorem 1, we know that $|\tilde{G}_i| \approx O(l_i)$, and that the $\tilde{G}_i$ are close to zero for all but a leading number of coefficients. On the other hand, as discussed in Section 3.3, the transformed noise $\tilde{N}$ is typically evenly distributed over all coefficients. Thus, the coefficients of the noise have the shape of an evenly distributed "noise floor" $\tilde{N}_i$ from which the coefficients $\tilde{G}_i$ of the relevant information arise (see Figure 1(b) for an example).

The idea is now to find a cut-off point such that the coefficients are divided into two parts $z_1, \ldots, z_d$ and $z_{d+1}, \ldots, z_n$ such that the first part contains the relevant information and the latter part consists of evenly distributed noise. We model the coefficients by two zero-mean Gaussians with

individual variances

$$z_i \sim \begin{cases} \mathcal{N}(0,\sigma_1^2) & 1 \le i \le d, \\ \mathcal{N}(0,\sigma_2^2) & d < i \le n. \end{cases}$$

Of course, in order to be able to extract meaningful information, it should hold that $\sigma_1 \gg \sigma_2$. Alternatively, one could assume that $z_i \sim \mathcal{N}(0,\sigma_1^2+\sigma_2^2)$, for $1 \le i \le d$, which nevertheless leads to the exact same choice of d.

For real data, both parts need not be actually Gaussian distributed. However, due to lack of additional a priori knowledge on the signal or the noise, the Gaussian distribution represents the optimal choice among all distributions with the same variance according to the maximum entropy principle (Jaynes, 1957).

The negative log-likelihood is proportional to

$$-\log \ell(d) \sim \frac{d}{n}\log\sigma_1^2 + \frac{n-d}{n}\log\sigma_2^2, \qquad \text{with} \qquad \sigma_1^2 = \frac{1}{d}\sum_{i=1}^{d} z_i^2,\ \sigma_2^2 = \frac{1}{n-d}\sum_{i=d+1}^{n} z_i^2. \tag{3}$$

The estimated dimension is then given as the maximum likelihood fit

$$\hat{d} = \operatorname*{argmin}_{1\le d\le n'}(-\log \ell(d)) = \operatorname*{argmin}_{1\le d\le n'}\left(\frac{d}{n}\log\sigma_1^2 + \frac{n-d}{n}\log\sigma_2^2\right). \tag{4}$$

Due to numerical instabilities of kernel PCA components corresponding to small eigenvalues, the choice of d should be restricted to $1 \le d \le n' < n$: The coefficients z_i are computed by taking scalar products with eigenvectors u_i. For small eigenvalues (small meaning of the order of the available numerical precision, for double precision floating point numbers, this is typically around 10^{-16}), individual eigenvectors cannot be computed accurately, although the space spanned by all these eigenvectors is accurate. Therefore, coefficients z_i for large i are not be reliable. To systematically stabilize the algorithm, one should therefore limit the range of possible effective dimensions. We have found the choice of $1 \le d \le n/2$ to work well as this choice ensures that at least half of the coefficients are interpreted as noise. For very small and very complex data sets, this choice might prove suboptimal and better thresholds based, for example, on the actual decay of eigenvalues might be advisable. However, on all data sets discussed in this paper, the above choice performed very well.

4.1.2 RDE BY LEAVE-ONE-OUT CROSS-VALIDATION (LOO-CV)

We propose a second algorithm which is based on cross validation, a more general concept than parametric noise modeling. This algorithm only depends on our theoretical results to the extent that it searches for subspaces spanned by leading kernel PCA components. We later compare the two methods to see whether our assumptions were justified.

As stated in Lemma 1, the projection of Y onto the space spanned by the d leading kernel PCA components is given by $\sum_{i=1}^{d} u_i u_i^\top Y$, where u_i are the eigenvectors of the kernel matrix. The matrix $\mathbf{S} = \sum_{i=1}^{d} u_i u_i^\top$ can be interpreted as a "hat matrix" in the context of regression.[4] The idea is now to choose the dimension which minimizes the leave-one-out cross-validation error. This subspace then captures all of the relevant information about Y without overfitting.

4. Recall that for regression methods where the fitted function depends linearly on the labels, the matrix $\mathbf{S}$ which computes $\hat{Y} = \mathbf{S}Y$ is called the "hat matrix" since it "puts the hat on Y."

Computationally, note that one can write the squared error leave-one-out cross-validation in closed form, similar to kernel ridge regression (see Wahba, 1990):

$$\mathrm{cv}_{\mathrm{loo}}(d) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{[\mathbf{S}Y]_i - Y_i}{1-\mathbf{S}_{ii}}\right)^2.$$

It is possible to organize the computation in a way such that given the eigendecomposition of $\mathbf{K}$, each value $\mathrm{cv}_{\mathrm{loo}}(d)$ can be computed in $O(n)$ (instead of $O(n^2)$ if one naively implements the above formula): Note that $\mathbf{S}_{ii}$ is equal to $\sum_{j=1}^{d}(u_j)_i^2$, therefore, one can compute $\mathbf{S}_{ii}$ iteratively by

$$\begin{aligned}\mathbf{S}_{ii}^0 &\leftarrow 0\\ \mathbf{S}_{ii}^{d+1} &\leftarrow \mathbf{S}_{ii}^d + (u_{d+1})_i^2.\end{aligned}$$

In the same way, since $\hat{Y} = \mathbf{S}Y = \sum_{j=1}^{d} u_j u_j^\top Y$, we get that

$$\begin{aligned}\hat{Y}^0 &\leftarrow 0\\ \hat{Y}^{d+1} &\leftarrow \hat{Y}^d + u_{d+1}u_{d+1}^\top Y.\end{aligned}$$

The squared error is in principle not the most appropriate loss function for classification problems. But as we will see below, it nevertheless works well also for classification problems.

4.2 Denoising the Labels and Estimating the Noise Level

One direct application of the dimensionality estimate is the projection of Y onto the first $\hat{d}$ kernel PCA components. By Lemma 1, this projection is

$$\hat{G}' = \sum_{i=1}^{\hat{d}} u_i u_i^\top Y.$$

Then, an estimate of the noiseless labels is given by

$$\hat{G} = \begin{cases}\operatorname{sign}\hat{G}' & \text{classification against } \pm 1 \text{ labels}\\ \hat{G}' & \text{regression}\end{cases}. \tag{5}$$

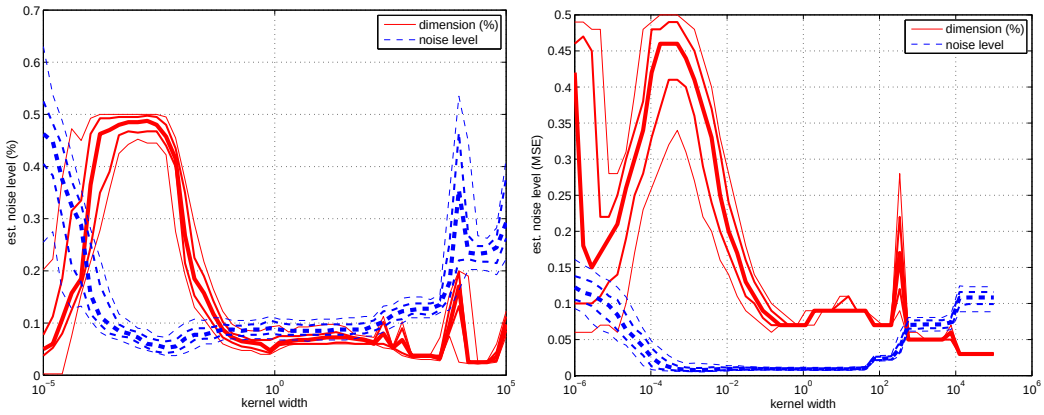
Note that this amounts to computing the in-sample fit using kernel principal component regression (kPCR).

The estimated dimension can also be used to estimate the noise level present in the data set by

$$\hat{\mathrm{err}} = \frac{1}{n}\sum_{i=1}^{n} L(\hat{Y}_i, Y_i), \tag{6}$$

where L is the loss function.

The accuracy of both these estimates depends on a number of factors. Basically, the estimation error is small if the first $\hat{d}$ kernel PCA components capture most of G and $\hat{d}$ is small such that most of the noise is removed. Note that our assumption that the kernel suits the data set is crucial for both these requirements. If g does not lie in the span of the associated integral operator T_k, the coefficients decay only slowly and a huge number of dimensions are necessary to capture most information about G, leading to a huge amount of residual noise.



(a) Classification ("banana" data set) (b) Regression (noisy sinc function)

Figure 5: Dimensions and estimated noise levels for varying kernel widths are not suited for model selection as it is unclear how to combine both estimates and they become instable for very small kernel widths. Shown are the 10%, 25%, 50%, 75%, and 90% percentiles over 100 resamples. **Legend:** "dimension (%)"—estimated dimensionality divided by number of samples. "noise level"—estimated noise level using the ℓ_1-norm for classification, and the (unnormalized) ℓ_2-norm for regression.

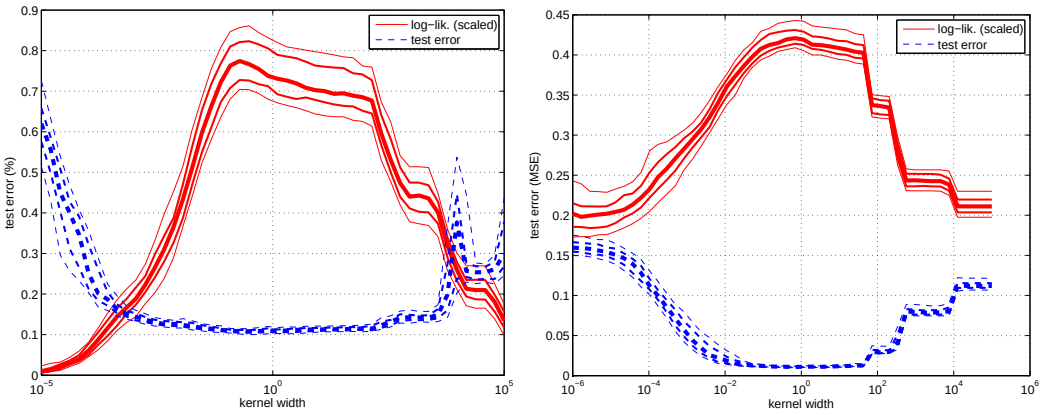
4.3 Applications to Model Selection

A highly relevant problem in the context of kernel methods is the selection of a kernel from a number of possible candidates which fits the problem best. This problem is usually solved by extensive cross-validation.

We would like to discuss possibilities to use the estimates introduced so far for model selection. Choosing the model based on either dimensionality or noise level alone is not sufficient, since one wants to optimize a combination of both. However, as the two terms live on quite different scales, it is unclear how to combine them effectively. Furthermore, as we will see below, both estimates alone become unstable for very small or very large kernel widths. The log-likelihood which achieves the optimum in (4) overcomes both problems and can be used for effective model selection.

Let us first discuss how the relation between the scale of the kernel and the data set can affect the dimensionality of the embedding in feature space. The standard example for a family of kernels with a scale parameter is the rbf-kernel (also known as Gaussian kernel, see Appendix A). Figure 5 shows the dimension and noise level estimates for a classification data set (the "banana" data set), and a regression data set (the "noisy sinc function" with 100 data points for training, and 1000 data points for testing) over a range of kernel widths. Generally speaking, if the scale of the kernel is too coarse for the problem, the problem tends to appear to be very low-dimensional with a large amount of noise. On the other hand, if the scale of the kernel is too fine the learning problem appears to be very complex with almost no noise.

Now, the log-likelihood $\ell(\hat{d})$ solves both problems. It combines the dimension and the noise level into a single meaningful number, and its value is stable across the whole scale range. In Figure 6, we have plotted the log-likelihood (scaled to fit into the plot) against the test error, both



(a) Classification ("banana" data set) (b) Regression (noisy sinc function)

Figure 6: Comparison of test errors and the negative log-likelihood from Equation (3) shows that the negative log-likelihood is highly correlated with the test error and can thus be used for model selection. Shown are the 10%, 25%, 50%, 75%, and 90% percentiles over 100 resamples. **Legend:** "log-lik. (scaled)"—log-likelihood (scaled). "test error"—test error using the ℓ_1-norm for classification, or the (unnormalized) ℓ_2-norm for regression.

with respect to the classification and least squares error. We see that the estimated log-likelihoods can be estimated well over the whole range, and that the likelihoods are highly correlated with the actual test error. Thus, the log-likelihood is a reliable indicator for the test errors based on the best separation between signal and noise.

Another alternative, which is somewhat more straight-forward, but conceptually also less interesting, is to use the leave-one-out cross-validation error. This quantity also measures how well the kernel can separate the noise from the relevant information, and is directly linked to the test error on an independent data set. We validate both model selection approaches experimentally in Section 5.

4.4 Applications to Data Set Assessment

When working on a concrete data set in a kernel setting, one is faced with the problem of finding a suitable kernel. This problem is usually approached with a mix of hard-won experience and domain knowledge. The main tool for guiding the search are prediction performance measures, the classical one being prediction accuracy. Measurements like the ROC (receiver-operator-curve), or the AUC (area-under-the-curve) give more fine-grained measurements of prediction quality, in particular in areas where many false positives or false negatives are not acceptable.

If, after testing a number of sensible candidates, the achieved prediction quality is satisfying, this approach is perfectly adequate, but more often than not, prediction quality is not as good as desirable. In such a case, it is important to identify the cause for the inferior performance. In principle, three alternatives are possible:

1. The kernels which have been used so far are not suited for the problem.

2. The learning problem is very complex and requires more data.

data set	RDE method	dimension	noise-level
complex data set	TCM	50	16.07%
	LOO-CV	25	40.59%
noisy data set	TCM	9	40.71%
	LOO-CV	9	40.71%

Table 2: Estimated dimensions for the two data sets from Figure 7. Methods are "TCM" for RDE by fitting a two-component model, "LOO-CV" for RDE by leave-one-out cross-validation. "noise-level" is measured as normalize mean square error (see Appendix A).

3. Better performance cannot be achieved since the learning problem is intrinsically noisy.

Each of these alternatives requires different approaches. In the first case, a better kernel has to be devised, in the second case, more data has to be acquired, and in the last case, one can either stop searching for a better kernel, or try to improve the quality of the data or the features used.

Ultimately, these questions cannot be answered without knowledge of the true distribution of the data, but the important observation here is that performance measures do not provide enough information to distinguish these cases.

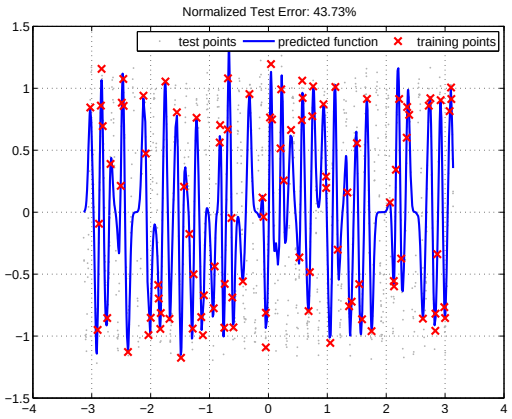
The estimates introduced so far can now be used to obtain evidence for distinguishing between the second and third case. On the one hand, the dimensionality of the problem is related to the complexity of the problem, while the noise level measures the inherent noise. Note that both these estimates depend on the chosen kernel.

Consider the following example: We study two data sets, a simple data set built from a noisy sinc function, and a complex data set based on a high-frequency sine function (see Figure 7). For the same number of data points $n = 100$, both data sets lead to comparable normalized test errors[5] for the best model selected (A normalized test error of 43.7% on the complex data set and 44.4% on the noisy data set using kernel ridge regression with model selection by leave-one-out cross-validation. Widths were selected from 20 logarithmically spaced points from 10^{-6} to 10^{2}, regularization constant was selected from 10 logarithmically spaced points from 10^{-6} to 10^{3}). However, the reason for the large error on the complex data set is clearly due to the small number of samples. If we increase the data set size to 1000 points, the normalized test error becomes 2.4%.

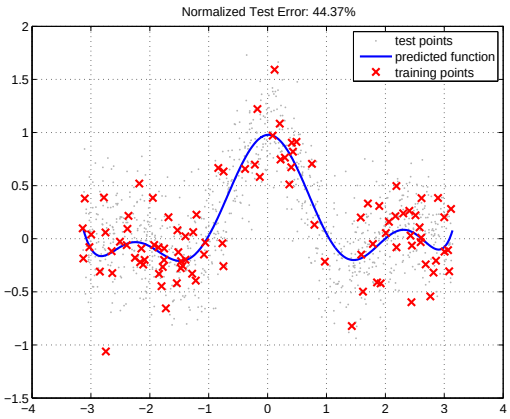
The question is now whether we can distinguish these two cases based on the kernel PCA coefficients. In fact, even on visual inspection, the kernel PCA coefficients display significant differences (see Figures 7(c) and 7(d)). We estimate the effective dimension and the resulting noise-level using the two methods we have proposed, the results are shown in Table 2. While both methods lead to different estimates, they both agree on the fact that the noisy data set has comparably low complexity and high noise, while the complex data set is quite high-dimensional, in particular if one takes into account that the data set contains only 100 data points. In fact, the RDE analysis on the larger complex data set with 1000 data points gives a dimension of 142, and a noise-level of 1.96%. Thus, the RDE measure correctly indicates that the large test error is due to the insufficient amount of data in the one case, and due to the large noise level in the other case.

This simple example demonstrates how the RDE measure can provide further information beyond the error rates. Below, we discuss this approach for several benchmark data sets.

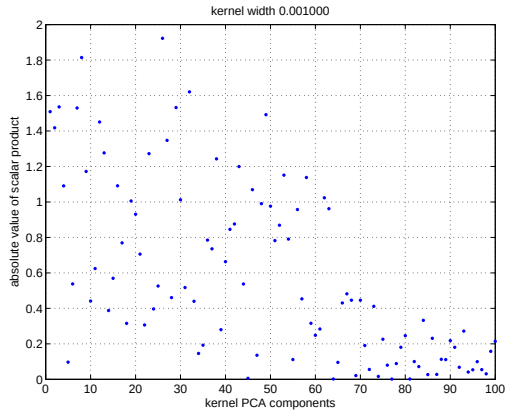
5. See Appendix A for a definition of the normalized error.



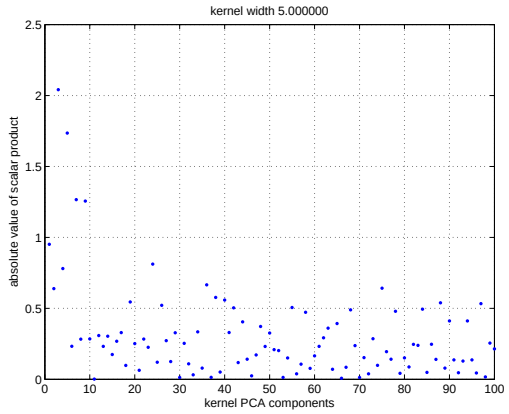
(a) A complex data set.



(b) A noisy data set.



(c) Kernel PCA coefficients for the complex data set.



(d) Kernel PCA coefficients for the noisy data set.

Figure 7: For both data sets, the X values were sampled uniformly between $-\pi$ and π. For the complex data set, $Y = \sin(35X) + \varepsilon$ where ε has mean zero and variance 0.01. For the noisy data set, $Y = \mathrm{sinc}(X) + \varepsilon'$ where ε' has mean zero and variance 0.09. Errors are reported as normalized mean squared error (see Appendix A). Below, the kernel PCA coefficients (scalar products with eigenvectors of the kernel matrix) for the optimal kernel selected based on the RDE (TCM) estimates are plotted. Coefficients are sorted by decreasing corresponding eigenvalue.

5. Experiments

We test our methods on several benchmark data sets. As discussed in the introduction, in order to validate whether our dimension estimates are accurate, we compare the achieved test error rates on the reduced feature space to other state-of-the-art algorithms. If the estimate is accurate, the test errors should be on par with these algorithms. Furthermore, we apply our method to estimate the complexity and noise level of the various data sets.

5.1 Benchmark Data Sets

We performed experiments on the classification data sets from Rätsch et al. (2001). For each of the data sets, we analyze it using a family of rbf kernels (see Appendix A). The kernel width is selected automatically using the achieved log-likelihood as described above. The width of the rbf kernel is selected from 20 logarithmically spaced points between 10^{-2} and 10^4 for each data set.

Table 3 shows the resulting dimension estimates using both RDE methods, with the cross-validation based RDE method being slightly biased towards higher dimensions. We see that both methods perform on par, which shows that the strong structural prior assumption underlying RDE is justified.

To assess the accuracy of the dimensionality estimate, we compare an unregularized least-squares fit in the reduced feature space (RDE+kPCR) with kernel ridge regression (KRR) and support vector machines (SVM) on the original data set. The resulting test errors are also shown in Table 3. Note that the combination of RDE and kPCR is conceptually very similar to the kernel projection machine (Vert et al., 2005) which also produces comparable results. However, in that paper, no practical method for estimating the dimension (beyond cross-validation) has been proposed. From the resulting test errors, we see that a relatively simple method on the reduced features performs on par with the state-of-the-art competitors. We conclude that the identified reduced feature space really contains all of the relevant information. Also note that the estimated noise levels match the actually observed error rates quite well, although there is a slight tendency to under-estimate the true error.

As discussed in Section 4.4, while the test errors only suggest a linear ordering of the data sets by increasing difficulty, using the dimension and noise level estimates, a more fine-grained analysis is possible. We can roughly divide the data sets into four classes (see Table 4), depending on whether the dimensionality is small or large, and the noise level is low or high. Data sets with small noise level show good results, almost irrespective of the dimensionality. The data set *image* seems to be particularly noise free, given that one can achieve a small error in spite of the large dimensionality.

The data sets *breast-cancer*, *diabetes*, *flare-solar*, *german*, and *titanic*, which all have test errors of 20% or more, have only moderately large dimensionalities. This means that the complexity of the underlying optimal decision boundary is not overly large (at least when viewed through the lens of the rbf-kernel), but a large inherent noise level prevents better results. Since this holds for rbf-kernels over a wide range of kernel widths, these results can be taken as a strong indicator that the Bayes error is in fact large.

The *splice* data set seems to be a good candidate for improvement. The noise level is moderately high, while the dimensionality with respect to the rbf-kernel seems quite high. We would like to use our dimensionality and noise level estimate as a tool to examine different kernel choices. (See Section C for further details).

Closer inspection of the data set reveals that a plain rbf-kernel is a suboptimal choice. The task of the splice data set consists in predicting whether there is a *splice-site* in the middle of a string of DNA (such sites encode the beginning and endings of coding regions on the DNA). In the data set, the four amino-acids A, C, G, T are encoded as numbers 1, 2, 3, and 4. Therefore, an rbf-kernel incorrectly assumes that C and G are more similar than A and T. One alternative which is more suited to this data set consists in encoding A, C, G, and T as binary four-vectors. The resulting kernel matrix has much smaller dimension, and also a smaller error rate (see Table 5).

data set	TCM	LOO-CV	TCM-noise level	RDE+kPCR	KRR	SVM
banana	24	26	8.8 ± 1.5	*11.3* $\pm$ *0.7*	**10.6** $\pm$ **0.5**	11.5 ± 0.7
breast-cancer	2	2	25.6 ± 2.1	27.0 ± 4.6	*26.5* $\pm$ *4.7*	**26.0** $\pm$ **4.7**
diabetes	9	9	21.5 ± 1.3	23.6 ± 1.8	**23.2** $\pm$ **1.7**	23.5 ± 1.7
flare-solar	10	10	32.9 ± 1.2	*33.3* $\pm$ *1.8*	34.1 ± 1.8	**32.4** $\pm$ **1.8**
german	12	12	22.9 ± 1.1	24.1 ± 2.1	**23.5** $\pm$ **2.2**	*23.6* $\pm$ *2.1*
heart	4	5	15.8 ± 2.5	16.7 ± 3.8	*16.6* $\pm$ *3.5*	**16.0** $\pm$ **3.3**
image	272	368	1.7 ± 1.0	4.2 ± 0.9	**2.8** $\pm$ **0.5**	*3.0* $\pm$ *0.6*
ringnorm	36	37	1.9 ± 0.7	*4.4* $\pm$ *1.2*	4.7 ± 0.8	**1.7** $\pm$ **0.1**
splice	92	89	9.2 ± 1.3	13.8 ± 0.9	*11.0* $\pm$ *0.6*	**10.9** $\pm$ **0.6**
thyroid	17	18	2.0 ± 1.0	5.1 ± 2.1	**4.3** $\pm$ **2.3**	*4.8* $\pm$ *2.2*
titanic	4	6	20.8 ± 3.8	22.9 ± 1.6	*22.5* $\pm$ *1.0*	**22.4** $\pm$ **1.0**
twonorm	2	2	2.3 ± 0.7	**2.4** $\pm$ **0.1**	*2.8* $\pm$ *0.2*	3.0 ± 0.2
waveform	14	23	8.4 ± 1.5	10.8 ± 0.9	**9.7** $\pm$ **0.4**	*9.9* $\pm$ *0.4*

Table 3: Estimated dimensions and error rates for the benchmark data sets from Rätsch et al. (2001). **Legend:** "TCM"—medians of estimated dimensionalities over resamples using the RDE by TCM methods. "LOO-CV"—dimensionality estimated by leave-one-out cross-validation. "TCM-noise level"—estimated error rate using the estimated dimension. "RDE+kPCR"—test error using a least-squares hyperplane on the estimated subspace in feature space. "KRR"—kernel ridge regression with parameters determined by leave-one-out cross-validation. "SVM"—the original error rates from Rätsch et al. (2001). **Best** and *second best* results are highlighted.

	low noise	high noise
low dimensional	banana, thyroid, waveform	breast-cancer, diabetes flare-solar, german heart, titanic
high dimensional	image, ringnorm	splice

Table 4: The data sets by noise level and complexity.

Still, there is further room for improvement. Using a *weighted-degree kernel*, which has been specifically designed for this problem (Sonnenburg et al., 2005), we obtain even better results: While the dimension is again slightly larger (but still moderate compared to the number of 1000 training examples), the noise level is even smaller. The reason is that the weighted degree kernel weights longer consecutive matches on the DNA differently while the rbf kernel just compares individual matches. Again, learning hyperplanes on the subspace of the estimated dimension leads to classification results on the test sets which are close to those predicted by the error level estimate.

6. Discussion

We discuss some implications of our results to learning theory. In particular we show how the "standard picture" on kernels and feature spaces is extended by our results. With respect to practical

kernel	RDE	est. error rate	RDE+kPCR
rbf	87	9.4 ± 1.0	12.9 ± 0.9
rbf (binary)	11	7.1 ± 1.0	7.6 ± 0.7
wdk	29	4.5 ± 0.7	5.5 ± 0.7

Table 5: Different kernels for the splice data set (for fixed kernel width $w = 50$). **Legend:** "rbf"—plain rbf-kernel, "rbf (binary)"—rbf-kernel on A, C, G, T encoded in binary four-vectors, "wdk"—weighted degree kernel (Sonnenburg et al., 2005).

applications we explain the role of RDE as a diagnosis tool for kernels. We close by contrasting our notion of dimension with two closely related dimensions, the dimension of the minimal subspace necessary to capture the relevant information about a learning problem, and the dimension of the data sub-manifold.

6.1 Connections to Learning Theory

We start with some informal reasoning about our findings much like in the spirit of Vapnik (1995). Although our ideas are not developed to all formal details, they are intended to provide some interesting insights on extensions to the general statistical learning theory picture (see Figure 8). The standard picture (see, for example, Burges, 1998; Müller et al., 2001) can be summarized as follows: The learning problem is given in terms of a finite data set in $\mathcal{X} \times \mathcal{Y}$. The kernel k implicitly embeds $\mathcal{X}$ in some (potentially) high-dimensional feature space $\mathcal{F}$ via the feature map Φ. Now since the feature space can be high-dimensional, it is argued that one needs to employ some form of appropriate complexity control in order to be able to learn. A prominent example are large margin classifiers, leading to support vector machines. Other examples include penalization of the norm of the weight vectors, which relates to a penalization of the norm in the resulting reproducing kernel Hilbert space (RKHS).

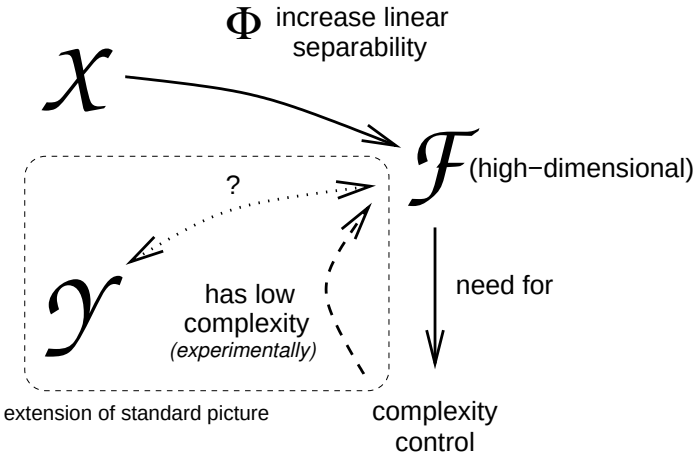


Figure 8: Learning in kernel feature spaces.

This picture is not entirely conclusive since it is not a priori clear that the feature map and the complexity control interact in a benign fashion. For example, it might be possible that the

feature map transforms the data such that a good representation can be learned, but the solution is incompatible with the kind of complexity one is penalizing. On the other hand, the large body of successful applications of kernel methods to real world problems is ample experimental verification of the fact that this seems to be the case and choosing a good kernel leads to an embedding which has low complexity, permitting, for example, large margin classifiers.

The question of the complexity of the image of $\mathcal{X}$ under the feature map actually has two parts. Part 1 concerns the complexity of the embedded object features $\Phi(X)$, while part 2 concerns the relation between the labels Y and the embedded object features $\Phi(X)$.

The first part has already been studied in several works. For example, Blanchard et al. (2007) and Braun (2006) have derived approximation bounds which show that the principal component values approximate the true principal values quickly (see also Mika, 2002; Shawe-Taylor et al., 2005). And since the asymptotic principal values decay rapidly, these results show that most of the variance of the X in feature space is contained in a finite dimensional subspace in feature space. Considering the function class generated by the feature map, Shawe-Taylor et al. (1998) first dealt with the complexity of kernel classes showing that the complexity can be bounded in the spirit of the structural risk minimization framework if a properly regularized class is picked depending on the data, for example by using large margin hyperplanes. Williamson et al. (2001) have further refined these results by using the concept of entropy numbers for compact operators that the complexity of the resulting hypothesis class is actually finite at any given positive scale. Evgeniou and Pontil (1999) show, using the concept of V_γ-dimension, which directly translates to a constraint on the RKHS-norm of the functions, that the resulting hypothesis classes have finite complexity. In summary, the embedding of $\mathcal{X}$ is known to have finite complexity (up to a small residual error).

The second part addresses the question if the embedding also relates favorably to the labels. In this work we have studied this question and answered it positively. One can prove that under mild assumptions on the general fit between the kernel and the learning problem, the information about the labels is always contained in the (typically small) subspace also containing most of the variance about the object features. While this borders on the trivial for the asymptotic setting, we could show that the same also holds true for a concrete finite data set, even at small sample sizes.

Our findings clarify the role of complexity control in feature space. The complexity control is not sufficient for effective learning in the feature space, but necessary. In conjunction with a sensible embedding provided by a suitable choice of the kernel function, it ensures that learning focuses on the relevant information and prevents overfitting. Interestingly, RKHS type penalty terms automatically ensure that the learned function focuses on directions in which the data has large variance, automatically leading to a concentration on the leading kernel PCA components.

6.2 RDE as a Diagnosis Tool

As discussed in Section 4.4, performance measures like the test error are very useful to compare different kernels, but fail to provide evidence if the performance is not as good as desired on whether the right kernel has not been found yet or the problem is intrinsically noisy.

Now, the RDE based estimates proposed in this paper offer a possible new approach to solve this problem. The relevant dimensionality estimate and the noise level estimate allow us to directly address the complexity vs. randomness issue, at least for a given kernel. Of course, our approach only provides a partial answer. However, using a generic kernel like an rbf-kernel for different widths results in an analysis of the data set on a whole range of scale resolutions. If the data set

appears to be low-dimensional and noisy at every scale, there is a strong indication that the noise level is actually quite high.

In the data sets discussed in Section 5, we have considered kernel widths in the range 10^{-2} to 10^4. The data sets *breast-cancer*, *diabetes*, *flare-solar*, *german*, *heart*, and *titanic*, which all have prediction errors larger than 15%, turn out to be fairly low-dimensional over the whole range.

On the other hand, the splice data set seemed to be quite complex, but not very noisy. Using domain knowledge, we improved the encoding, and finally chose a different kernel, which further reduced the complexity and noise (see Section C for further details).

In summary, using the RDE based estimates as a diagnosis tool, it is possible to obtain more detailed insights into how well a kernel is adapted to the characteristic properties of a data set and its underlying distribution than by using integrative performance measures like test errors only.

6.3 The "True" Dimensionality of the Data

We estimate the number of leading kernel PCA components necessary to capture the relevant information contained in the learning problem. This "relevant dimensionality estimate" captures only a very special kind of dimensionality notion, and we would like to compare it with two other aspects of dimensionality.

In our dimensionality estimate, the basis was fixed and given by leading kernel PCA components. One might wonder how many dimensions are necessary to capture the relevant information about the learning problem if one were also allowed to choose the basis. The answer is easy: In order to capture G, it suffices to consider the one-dimensional space spanned by G itself, which means that the minimal dimensionality of the learning problem is 1. However, note that G is not known, and estimating G amounts to solving the learning problem itself. In other words, the choice of a kernel can be interpreted as implicitly specifying an appropriate basis in feature space which is able to capture G using as few basis vector as possible, *and* also using a subspace which contains as much of the variance of the data as possible.

For most data sets, the different input variables are highly dependent, such that the data does not occupy all of the space but only a sub-manifold in the space. The dimension of this sub-manifold is a further notion of dimensionality of a data set. However, note that we consider the dimensionality of the data with respect to the information in the labels, while the sub-manifold view usually concentrates on the inputs only. Also, we are considering linear subspaces (in an RKHS), which typically require more dimensions to capture the data than a non-linear manifold would. On the other hand, since we are only looking at the subspace which is relevant for predicting the labels, the estimated dimension may also be smaller than the dimension of the data manifold in feature space.

7. Conclusion

Both in theory and on practical data sets, we have demonstrated that the relevant information in a supervised learning scenario is contained in the leading projected kernel PCA components if the kernel matches the learning problem and is sufficiently smooth. This behavior complements the common statistical learning theoretical view on kernel based learning adding insight on the intricate interplay of data and kernel: An appropriately selected kernel (a) leads to an efficient model which generalizes well, since only a comparatively low dimensional representation has to be learned for a

fixed given data size. An appropriately selected kernel (b) permits a dimension reduction step that discards some irrelevant projected kernel PCA directions and thus yields a regularized model.

We propose two algorithms for the relevant dimensionality estimate (RDE) task. These can also be used to automatically select a suitable kernel model for the data and to extract as additional side information an estimate of the effective dimension and estimated expected error for the learning problem. Compared to common cross-validation techniques one could argue that all we have achieved is to find a similar model as usual at a comparable computing time. However, we would like to emphasize that the side information extracted by our procedure contributes to a better understanding of the learning problem at hand: Is the classification result limited due to intrinsic high dimensional structure or are we facing noise and nuisance dimensions? Simulations show the usefulness of our RDE algorithms.

An interesting future direction lies in combining these results with generalization bounds which are also based on the notion of an effective dimension, this time, however, with respect to some regularized hypothesis class (see, for example, Zhang, 2005). Linking the effective dimension of a data set with the "dimension" of a learning algorithm, one could obtain data dependent bounds in a natural way with the potential to be tighter than bounds which are based on the abstract capacity of a hypothesis class.

Acknowledgments

Parts of this work have been performed while MLB was with the Intelligent Data Analysis Group at the Fraunhofer Institute FIRST. The authors would like to thank Volker Roth, Tilman Lange, Gilles Blanchard, Stefan Harmeling, Motoaki Kawanabe, Claudia Sannelli, Jan Müller, and Nicole Krämer for fruitful discussions. The authors would also like to thank the anonymous referees whose comments have helped to improve the paper further, and in particular Peter Bartlett for his valuable comments. This work was supported in part by the BMBF FaSor project, 16SV2234, and by the FP7-ICT Programme of the European Community, under the PASCAL2 Network of Excellence, ICT-216886.

Appendix A. Data Sets and Kernel Functions

In this section, we introduce some data sets and define the Gaussian kernel, since there exists some variability with respect to its parameterization.

A.1 Gaussian kernel

The Gaussian kernel, or rbf-kernel, used in this paper are parameterized as follows: The Gaussian with width w is

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2w}\right).$$

A.2 Classification Data Sets

For classification, we use the data sets from Rätsch et al. (2001). This benchmark data set consists of 13 classification data sets, which are partly synthetic, and partly derived from real-world data. The data sets are pre-arranged into different resamples of training and test data sets. The number

of resamples is 100 with the exception of the "image" and "splice" data sets which have only 20 resamples (because these data sets are fairly large). For visualization purposes, we often take the first resample of the "banana" data set, which is a two-dimensional classification problem (see Figure 1(a)).

A.3 Regression Data Sets

The "noisy sinc function" data set is defined as follows:

$$\begin{aligned} X_i &\sim \text{uniformly from } [-\pi,\pi], \\ Y_i &= \operatorname{sinc}(X_i) + \varepsilon_i, \\ \varepsilon_i &\sim \mathcal{N}(0,\sigma_\varepsilon^2). \end{aligned}$$

There are different alternatives for defining the sinc function, we choose $\operatorname{sinc}(x) = \sin(\pi x)/\pi x$, $\operatorname{sinc}(0) = 1$.

For regression, we sometimes measure the error using the "normalized mean squared error." If the original labels are given by Y_i, $1 \le i \le n$, and the predicted ones are $\hat{Y}_i$, then this error is defined as

$$\text{nmse} = \frac{\sum_{i=1}^n (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^n (Y_i - \frac{1}{n}\sum_{j=1}^n Y_j)^2}.$$

Appendix B. Proof of the Main Theorem

In this section, the main theorem of the paper is stated and proven. We start with some definitions, then introduce and discuss the assumptions of the main result. Next we define a few quantities on which the bound depends. The bound itself is split into two theorems. First the general bound is derived and then the asymptotic rates of these quantities are studied.

B.1 Preliminaries

Using the probability measure P_X which generates the Xs, we can define a scalar product via $\langle f,g\rangle = \int_X f(x)g(x)\mathrm{P}_X(dx)$ which induces the Hilbert space $\mathcal{L}^2(\mathcal{X},\mathrm{P}_X)$. Unless indicated otherwise, $\|f\|$ will denote the norm with respect to this scalar product. Let $k(x,y) = \sum_{\ell=1}^\infty \lambda_\ell \psi_\ell(x)\psi_\ell(y)$ be a kernel function (such that $\lambda_\ell \ge 0$). The ψ_ℓ form an orthogonal family of functions on the Hilbert space $\mathcal{L}^2(\mathcal{X},\mathrm{P}_X)$. Given an n-sample $X_1,\ldots,X_n$ from P_X, the sample vector of a function g is the vector $g(\mathbf{X}) = (g(X_1),\ldots,g(X_n))$. The kernel matrix given a kernel function k and an n-sample $X_1,\ldots,X_n$ is the $n\times n$ matrix $\mathbf{K}$ with entries $k(X_i,X_j)/n$.

Let $g(x) = \sum_{\ell=1}^\infty \alpha_\ell \lambda_\ell \psi_\ell(x)$ with $(\alpha_\ell) \in \ell^2$, the set of all square-summable sequences. The expansion of g in terms of $\lambda_\ell\psi_\ell$ amounts to assuming that g lies in the range of the integral operator T_k defined by $T_k f = \int_\mathcal{X} k(\cdot,x) f(x)\mathrm{P}_X(dx)$. Then, $g = T_k h$ with $h = \sum_{\ell=1}^\infty \alpha_\ell\psi_\ell$.

The act of truncating an object with an infinite expansion to its first r coefficients is so ubiquitous in the following that we introduce a generic notation. If k is a kernel function, $\tilde{k}$ is the kernel function whose expansion has been reduced to the first r terms. Likewise, $\tilde{\mathbf{K}}$ is the kernel matrix induced by $\tilde{k}$. For a sequence $(\alpha_\ell) \in \ell^2$, $\tilde{\alpha}$ is the tuple consisting of the first r elements of the sequence. The sample vector matrices $\tilde{\Psi}$ is formed by the sample vector of the first r eigenvectors, that is, $\tilde{\Psi}_{ij} = \psi_j(X_i)/\sqrt{n}$, and $\tilde{\Lambda}$ is the diagonal matrix formed from the first r eigenvalues, such that $\tilde{\mathbf{K}} = \tilde{\Psi}\tilde{\Lambda}\tilde{\Psi}^\top$. Finally, $\tilde{g}$ is obtained from g by truncating the expansion to the first r eigenfunctions.

The eigen-decompositions of the kernel matrix and the truncated kernel matrix (kernel matrix for the truncated kernel function) are

$$\mathbf{K} = \mathbf{U}\mathbf{L}\mathbf{U}^\top, \qquad \tilde{\mathbf{K}} = \tilde{\mathbf{U}}\tilde{\mathbf{L}}\tilde{\mathbf{U}}^\top,$$

where $\mathbf{U}$, $\tilde{\mathbf{U}}$ are orthogonal matrices with columns u_i, $\tilde{u}_j$, and $\mathbf{L}$, $\tilde{\mathbf{L}}$ are diagonal matrices with entries l_i, $\tilde{l}_j$, such that the eigenpairs of $\mathbf{K}$ are (l_i, u_i), and those of $\tilde{\mathbf{K}}$ are $(\tilde{l}_j, \tilde{u}_j)$. We stick to the general convention that eigenvalues are always sorted in decreasing order.

Tail-sums of eigenvalues are denoted by

$$\Lambda_{>r} = \sum_{i=r+1}^{\infty} \lambda_i, \qquad \Lambda_{\geq r} = \sum_{i=r}^{\infty} \lambda_i.$$

We will refer to the following result relating decay rates of the eigenvalues to the tail-sums. For proofs, see, for example, Braun (2006). It holds that if $\lambda_r = r^{-d}$ with $d \geq 1$, then $\Lambda_{>r} = O(r^{1-d})$. If $\lambda_r = \exp(-cr)$ with $c > 0$, then $\Lambda_{>r} = O(\exp(-cr))$. The same rates hold for $\Lambda_{\geq r}$.

Furthermore, we will often make use of the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ if $a, b \geq 0$.

B.2 Assumptions

The overall goal is to derive a meaningful upper bound on $\frac{1}{\sqrt{n}}|u_i^\top g(\mathbf{X})|$. In particular, the bound should scale with the corresponding eigenvalue l_i. We proceed as follows: First, we derive the actual bound which depends on a number of quantities. In the next step, we estimate the worst case asymptotic rates of these quantities. The actual bound depends on assumptions which are discussed in the following.

(A1) We assume that the kernel is uniformly bounded, that is,

$$\sup_{x,y \in \mathcal{X} \times \mathcal{X}} |k(x,y)| = K < \infty.$$

(A2) We assume that $n \geq r$ large enough such that $\tilde{\Psi}^\top \tilde{\Psi}$ is invertible.

(A3) We assume that $\lambda_i = O(i^{-5/2-\varepsilon})$ for some $\varepsilon > 0$.

Assumption (A1) is true for radial basis functions like the Gaussian kernel, but also if the underlying space $\mathcal{X}$ is compact and the kernel is continuous. From (A1), it follows easily that g is bounded as well since

$$|g(x)| \leq K\|h\|.$$

Furthermore, since the ψ_i are orthogonal, it follows that $\|h - \tilde{h}\| \leq \|h\|$, and therefore

$$|g(x) - \tilde{g}(x)| \leq K\|h\|$$

since $g - \tilde{g} = T_k(h - \tilde{h})$, and therefore $|g(x) - \tilde{g}(x)| \leq K\|h - \tilde{h}\| \leq K\|h\|$. These inequalities play an important role for bounding the truncation error $g - \tilde{g}$ in a finite sample setting.

Since the sample vectors $\psi_\ell(\mathbf{X})$ are asymptotically pairwise orthogonal, $\tilde{\Psi}^\top \tilde{\Psi}$ converges to $\mathbf{I}$, and for large enough n, assumption (A2) is met. See also Lemma 2 below.

Assumption (A3) ensures that the term $r(\sum_{i=1}^{r}|\alpha_i|)\Lambda_{\geq r}$ occurring in the bound vanishes as $r \to \infty$. Note that since the sequence of α_i is square-summable,

$$\sum_{i=1}^{r}|\alpha_i| \leq \sqrt{r\sum_{i=1}^{r}\alpha_i^2} \leq \sqrt{r}\|\alpha\|_{\ell^2} = O(\sqrt{r}).$$

Therefore, $r\sum_{i=1}^{r}|\alpha_i| = O(r^{3/2})$. Also, $\Lambda_{\geq r} = O(r^{-3/2-\varepsilon})$, such that $r(\sum_{i=1}^{r}|\alpha_i|)\Lambda_{\geq r} = O(r^{-\varepsilon})$. Note that (A3) is quite modest and eigenvalues often decay much faster, even at exponential rates.

B.3 The Main Result

The following five quantities occur in the bound:

- $c_i = |\{1 \leq j \leq r \mid l_i/2 \leq \tilde{l}_j \leq 2l_i\}|$ is the number of eigenvalues of the truncated kernel matrix which are close to the eigenvalues of the normal kernel matrix. This is a measure for the approximate degeneracy of eigenvalues.
- $\tilde{a} = \|\tilde{\alpha}\|_1$, is a measure for the size of the first r coefficients which define g.
- $\tilde{\mathbf{E}} = \mathbf{K} - \tilde{\mathbf{K}}$ is the truncation error for the kernel matrix.
- $\tilde{T} = \|g - \tilde{g}\| = \sqrt{\sum_{j=r+1}^{\infty}\alpha_j^2\lambda_j^2}$ is the asymptotic truncation error for the function g.
- $F = \|g\|_\infty < \infty$, an upper bound on g.

We study these quantities in more detail after proving the actual bound, which follows next.

Theorem 1 *With the definitions introduced so far, it holds that with probability larger that* $1-\delta$, *for all* $1 \leq i \leq n$,

$$\frac{1}{\sqrt{n}}|u_i^\top g(\mathbf{X})| < \min_{1\leq r\leq n}\left[l_i c_i D(r,n) + E(r,n) + T(r,n)\right]$$

where the three terms are given by

$$D(r,n) = 2\tilde{a}\|\tilde{\Psi}^+\|, \qquad E(r,n) = 2r\tilde{a}\|\tilde{\Psi}^+\|\|\tilde{\mathbf{E}}\|, \qquad T(r,n) = \tilde{T} + \sqrt{F\tilde{T}}\sqrt[4]{\frac{1}{n\delta}}.$$

Proof First, we replace $g = \tilde{g} + (g - \tilde{g})$ and obtain

$$\frac{1}{\sqrt{n}}|u_i^\top g(\mathbf{X})| \leq \frac{1}{\sqrt{n}}|u_i^\top \tilde{g}(\mathbf{X})| + \frac{1}{\sqrt{n}}\|g(\mathbf{X}) - \tilde{g}(\mathbf{X})\| =: \text{(I)},$$

using the Cauchy-Schwarz-inequality and the fact that $\|u_i\| = 1$ for the second term.

Next, we re-express $\tilde{g}(\mathbf{X}) = \sum_{\ell=1}^{r}\lambda_\ell\alpha_\ell\psi_\ell(\mathbf{X})$ as follows. By definition, $\tilde{g}(\mathbf{X})$ lies in the image of $\tilde{\mathbf{K}}$, therefore, $\tilde{g}(\mathbf{X}) = \sum_{j=1}^{r}\tilde{u}_j\tilde{u}_j^\top\tilde{g}(\mathbf{X})$. Using both these equations, we obtain

$$\frac{1}{\sqrt{n}}|u_i^\top\tilde{g}(\mathbf{X})| \leq \sum_{\ell=1}^{r}|\alpha_\ell|\sum_{j=1}^{r}(u_i^\top\tilde{u}_j)\left[\frac{1}{\sqrt{n}}\lambda_\ell\psi_\ell(\mathbf{X})^\top\tilde{u}_j\right] =: \text{(II)}$$

The term $u_i^\top \tilde{u}_j$ measures the angle between the eigenvectors of $\mathbf{K}$ and $\tilde{\mathbf{K}}$. Note that $\mathbf{K}$ can be considered an additive perturbation of $\tilde{\mathbf{K}}$ by $\tilde{\mathbf{E}} = \mathbf{K} - \tilde{\mathbf{K}}$. Such perturbations are studied by the so-called *sin-theta-theorems*. Specializing Theorem 6.2 of Davis and Kahan (1970) (see Section D) to two single eigenvectors, we obtain that

$$|u_i^\top \tilde{u}_j| \leq \min\left(\frac{\|\tilde{\mathbf{E}}\|}{|l_i - \tilde{l}_j|}, 1\right).$$

The term $\lambda_\ell \psi_\ell(\mathbf{X})^\top \tilde{u}_j/\sqrt{n}$ is bounded by $\tilde{l}_j \|\tilde{\Psi}^+\|$ (where Ψ^+ denotes the pseudo-inverse of Ψ), since

$$\tilde{l}_j \tilde{u}_j = \tilde{\mathbf{K}}\tilde{u}_j = \tilde{\Psi}\tilde{\Lambda}\tilde{\Psi}^\top \tilde{u}_j \qquad \Rightarrow \qquad \tilde{l}_j \tilde{\Psi}^+ \tilde{u}_j = \tilde{\Lambda}\tilde{\Psi}^\top \tilde{u}_j.$$

Taking norms, we obtain $\|\tilde{\Lambda}\tilde{\Psi}^\top \tilde{u}_j\| \leq \tilde{l}_j \|\tilde{\Psi}^+\|$, from which the claimed inequality follows for each individual coordinate of the vector on the left-hand side.

Combining the bounds for the two terms $u_i^\top \tilde{u}_j$ and $\lambda_\ell \psi_\ell(\mathbf{X})^\top \tilde{u}_j/\sqrt{n}$, we obtain

$$(u_i^\top \tilde{u}_j)\left[\frac{1}{\sqrt{n}}\lambda_\ell \psi_\ell(\mathbf{X})^\top \tilde{u}_j\right] \leq \|\tilde{\Psi}^+\| \min\left(\frac{\|\tilde{\mathbf{E}}\|}{|l_i - \tilde{l}_j|}, 1\right)\tilde{l}_j =: \|\tilde{\Psi}^+\| c_{ij}.$$

For $j \notin J(l_i) = \{1 \leq j \leq r \mid \frac{1}{2} l_i \leq \tilde{l}_j \leq 2l_i\}$, it holds that $\|\tilde{\mathbf{E}}\|\tilde{l}_j/|l_i - \tilde{l}_j| \leq 2\|\tilde{\mathbf{E}}\|$, therefore,

$$\sum_{j=1}^{r} c_{ij} = \sum_{j \in J(l_i)} c_{ij} + \sum_{j \notin J(l_i)} c_{ij} \leq 2|J(l_i)|l_i + 2r\|\tilde{\mathbf{E}}\|.$$

We have just shown that

$$\text{(II)} \leq \|\tilde{\Psi}^+\| \sum_{\ell=1}^{r} |\alpha_\ell| \left(2|J(l_i)|l_i + 2r\|\tilde{\mathbf{E}}\|\right). \tag{7}$$

Now concerning the other term in (I), note that by the strong law of large numbers,

$$\frac{1}{n}\|g(\mathbf{X}) - \tilde{g}(\mathbf{X})\|_{\mathbb{R}^n}^2 \to \|g - \tilde{g}\|_{L^2(X, \mathrm{P}_X)}^2 = \sum_{j=r+1}^{\infty} \alpha_j^2 \lambda_j^2 =: \tilde{T}^2.$$

Since g is bounded, $\|g\|_\infty = F < \infty$, we can bound the variance of $g - \tilde{g}$:

$$\mathrm{Var}_{\mathrm{P}_X}((g - \tilde{g})^2) \leq \|g - \tilde{g}\|_\infty^2 \|g - \tilde{g}\|^2 = F^2 \tilde{T}^2.$$

We can thus bound the probability of a large deviation using the Chebychev-inequality. Taking the square roots, we obtain that with probability larger than $1 - \delta$,

$$\frac{1}{\sqrt{n}}\|g(\mathbf{X}) - \tilde{g}(\mathbf{X})\| \leq \tilde{T} + \sqrt{F\tilde{T}}(n\delta)^{-\frac{1}{4}}. \tag{8}$$

Combining bound (7) and (8), we obtain that

$$\frac{1}{\sqrt{n}}|u_i^\top g(\mathbf{X})| \leq 2l_i|J(l_i)|\|\tilde{\alpha}\|_1\|\tilde{\Psi}^+\| + 2r\|\tilde{\mathbf{E}}\|\|\tilde{\alpha}\|_1\|\tilde{\Psi}^+\| + \tilde{T} + \sqrt{F\tilde{T}}(n\delta)^{-\frac{1}{4}}.$$

This proves the upper bound on the coefficients. ■

B.4 Worst Case Asymptotic Rates of the Error Matrices

The bound depends on a number of error terms, whose worst case asymptotic rates and their dependency on r are studied next.

The norm of the pseudo-inverse of $\tilde{\Psi}$ can be related to the matrix $\tilde{\mathbf{C}} = \tilde{\Psi}^\top\tilde{\Psi} - \mathbf{I}$, which measures the deviation from orthonormality of the sample vectors of the first r eigenfunctions of T_k. Since the eigenfunctions are asymptotically orthonormal, it is guaranteed that $\|\tilde{\mathbf{C}}\| \to 0$ as $n \to \infty$.

Lemma 2 *Let* $\tilde{\mathbf{C}} = \tilde{\Psi}^\top\tilde{\Psi} - \mathbf{I}$. *If* $\|\tilde{\mathbf{C}}\| < 1$, *then*

$$\|\tilde{\Psi}^+\| \leq (1 - \|\tilde{\mathbf{C}}\|)^{-1/2} = 1 + O(\sqrt{\|\tilde{\mathbf{C}}\|}).$$

Proof Recall that $\|\tilde{\Psi}^+\| = 1/\sigma_r(\tilde{\Psi})$, where $\sigma_r(\tilde{\Psi})$ is the rth singular value of $\tilde{\Psi}$ in descending order. The singular values are the square roots of the eigenvalues of $\tilde{\Psi}^\top\tilde{\Psi}$, and

$$1 - \lambda_r(\tilde{\Psi}^\top\tilde{\Psi}) \leq \max_{1\leq i\leq r} |\lambda_i(\tilde{\Psi}^\top\tilde{\Psi}) - 1| \leq \|\tilde{\Psi}^\top\tilde{\Psi} - \mathbf{I}\|,$$

and therefore $\sigma_r(\tilde{\Psi}) = (\lambda_r(\tilde{\Psi}^\top\tilde{\Psi}))^{1/2} \geq (1 - \|\tilde{\Psi}^\top\tilde{\Psi} - \mathbf{I}\|)^{1/2}$, which proves the inequality.

For the asymptotic rate, observe that

$$\|\tilde{\Psi}^+\| \leq \sqrt{\frac{1}{1 - \|\tilde{\mathbf{C}}\|}} = \sqrt{\frac{\|\tilde{\mathbf{C}}\|^{-1}}{\|\tilde{\mathbf{C}}\|^{-1} - 1}} = \sqrt{1 + \frac{1}{\|\tilde{\mathbf{C}}\|^{-1} - 1}} \leq 1 + \sqrt{\frac{1}{\|\tilde{\mathbf{C}}\|^{-1} - 1}}.$$

Now, $1/(x-1) = O(1/x)$ for $x \to \infty$, which proves the asymptotic rate. ■

The two error matrices $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{E}}$ were discussed in depth by Braun (2006). However, note that these asymptotic rates are worst case rates over certain families of kernel functions. This means that the results on the asymptotic rates do not describe typical behavior but rather worst case behavior, and their main purpose of these rates is to ensure that the error terms cannot diverge rather than giving realistic estimates.

The following result is Theorem 4 from Braun (2006).

Lemma 3 *For* $1 \leq r \leq n$, *with probability larger than* $1 - \delta$,

$$\|\tilde{\mathbf{C}}\| < r\sqrt{\frac{r(r+1)K}{\lambda_r n\delta}}, \qquad \|\tilde{\mathbf{E}}\| < \lambda_r + \Lambda_{>r} + \sqrt{\frac{2K\Lambda_{>r}}{n\delta}}.$$

¿From Lemma 3, it follows that

$$\|\tilde{\Psi}^+\| = 1 + O(r\lambda_r^{-1/4} n^{-1/4}),$$
$$\|\tilde{\mathbf{E}}\| = \Lambda_{\geq r} + O(\sqrt{\Lambda_{>r}}\, n^{-1/2}).$$

If we plug these rates into the bound from Theorem 1 and suppress all parts which converge to zero, the bound becomes

$$\frac{1}{\sqrt{n}} |u_i^\top g(\mathbf{X})| \leq 2c_i\tilde{a}l_i + 2r\tilde{a}\Lambda_{\geq r} + \tilde{T} + \text{terms which vanish as } n \to \infty.$$

We see that the general structure of the bound consists a part which scales with the eigenvalue under consideration and an additive part which is independent of i. The factor of the scaling part increases with r since $\tilde{a} = O(\sqrt{r})$ in the worst case. At the same time, the truncation error $\tilde{T}$ arising from the truncation of g becomes smaller as r is increased, and by assumption (A3), it is ensured that the second term actually converges to zero as $r \to \infty$. The two parts therefore form a trade-off and by choosing r, one can balance these two terms.

Now, in particular the convergence of $\|\tilde{\Psi}^+\| \to 1$ can be quite slow in the worst case, if the eigenvalues of the kernel matrix decay quickly (see the paper by Braun, 2006, for a more thorough discussion including an artificial example of a kernel function which achieves the described rate). However, note that $\|\tilde{\Psi}^+\|$ only occurs in conjunction with terms involving eigenvalues, such that the overall bound still converges. For example, one can prove that a decay rate of λ_r faster than $O(r^{-12})$ ensures that $E(r,n) = 2r\tilde{a}\|\tilde{\Psi}^+\|\|\tilde{\mathbf{E}}\| \to 0$ for $r \to \infty$ independently of n: It holds that

$$E(r,n) = 2r\tilde{a}\|\tilde{\Psi}^+\|\|\tilde{\mathbf{E}}\| = 2r\tilde{a}\left(1 + O(r\lambda_r^{-1/4}n^{-1/4})\right)\left(\Lambda_{\geq r} + O(\sqrt{\Lambda_{>r}}n^{-1/2})\right).$$

If one expands the product, the term which decays slowest with respect to r is (recall that $\tilde{a} = O(\sqrt{r})$)

$$2r\tilde{a}O(r\lambda_r^{-1/4})O(\sqrt{\Lambda_{>r}}) = O(r^{5/2}\lambda_r^{-1/4}\Lambda_{>r}^{1/2}).$$

Now if $\lambda_r = r^{-d}$, then $\Lambda_{>r} = O(r^{1-d})$, and

$$O(r^{5/2}\lambda_r^{-1/4}\Lambda_{>r}^{1/2}) = O(r^{5/2}r^{d/4}r^{(1-d)/2}) = O(r^{3-d/4}).$$

We require that the exponent is smaller than 0 which is true if $d > 12$. Again, since these are worst case considerations, and usually r and n will be coupled in some way, the additive terms will be controlled even for slower decay rates.

An interesting feature of the bound is that it is uniform in i, which means that the bound holds simultaneously for all eigenvectors. Therefore, the individual bounds can be combined, for example, to sums of scalar products without a decrease in the probability with which the bound holds.

In principle, it is possible to further relate the decay rate of the eigenvalues of the kernel matrix l_i to the asymptotic eigenvalue λ_i, for example using bounds for individual eigenvalues (Braun, 2006), or tail-sums of eigenvalues (Blanchard et al., 2007; Shawe-Taylor et al., 2005) if we wish to explicitly control the component of the relevant information vector which is not contained in the leading kernel PCA directions.

Appendix C. A Worked Through Example

In this section, we work through the "splice" data set to show how one would perform a kernel fitness analysis using the methods presented here. The computations of the estimates proposed in Section 4 are summarized in Algorithm 1.

We start out with the splice data set. As explained in the main section, each data points encodes sequence of aminoacids. In the positive examples, there exists a so-called splice site in the center of the encoded DNA signal. The task requires to predict splice sites in these short DNA sequences.

Usually, one would start with some specific kernel, for example an rbf-kernel, train some kernel learning algorithm using this kernel, evaluate the kernel on some test data set, and start to select different parameters. There are two potential drawbacks following this approach: (1) there exists

Algorithm 1 Computing the estimates from Section 4

Input: Kernel matrix $\mathbf{K}$, label vector Y, loss function L
Output: kernel PCA coefficients z, dimensionality $\hat{d}$,
negative log-likelihood $\hat{\ell}$, denoised labels $\hat{Y}$,
noise-level $\hat{\text{err}}$

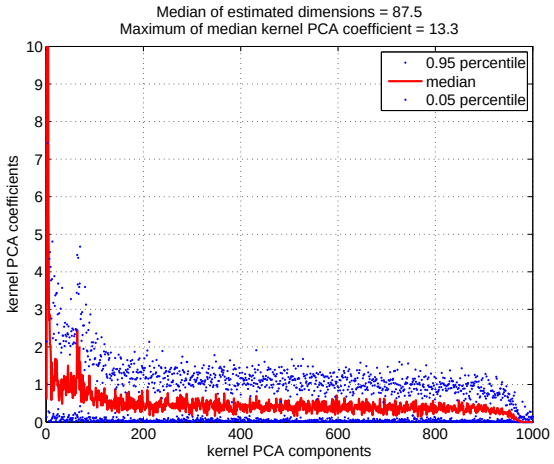
1: {*Compute kernel PCA coefficients*}
2: Compute eigendecomposition $\mathbf{KU} = \mathbf{UD}$
3: $z \leftarrow \mathbf{U}^\top Y$
4: {*Estimate dimensionality $\hat{d}$ (Eq. 4)*}
5: $c \leftarrow 0$; $C \leftarrow \|z\|^2$ {here, it is shown in detail how to achieve linear run-time}
6: **for** $d = 1$ to $n/2$ **do**
7: $\quad c \leftarrow c + z_i^2$
8: $\quad s_1 \leftarrow c/d$
9: $\quad s_2 \leftarrow (C-c)/(n-d)$
10: $\quad l_d \leftarrow d \log s_1 + (n-d)\log s_2$
11: **end for**
12: $\hat{d} \leftarrow \operatorname{argmin}_{1 \le d \le n/2} l_d$
13: $\hat{\ell} \leftarrow l_{\hat{d}}$.
14: {*Compute denoised labels (Eq. 5)*}
15: Extract first $\hat{d}$ eigenvectors $\mathbf{T} \leftarrow \mathbf{U}_{:,1:\hat{d}}$
16: $\hat{Y} \leftarrow \mathbf{TT}^\top Y$
17: {*Estimate noise-level (Eq. 6)*}
18: $\hat{err} = \frac{1}{n}\sum_{i=1}^n L(Y, \hat{Y})$

no absolute measure of the goodness of a certain kernel choice, only comparisons to other kernels, (2) there exists some dependency on the kernel learning method employed. Using the methods developed in this paper, it is possible to explore the relationship between the kernel and the data set in an algorithm independent way. Furthermore, in the case of poor performance, it is possible to distinguish between very complex cases (which require more input data), and cases where the data set appears to be very noisy (either requiring better data quality, or a kernel which can capture more information about the learning problem).

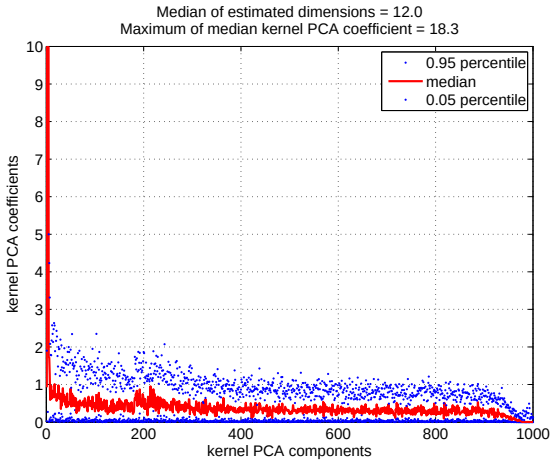
The splice data set consists of 20 resamples. We first try an rbf-kernel with width $w = 50$ (see Section A). We start by computing and plotting the kernel PCA coefficients. The resulting coefficients are plotted in Figure 9(a). We see that the data set appears to be rather high-dimensional, and the noise level is also quite high. The estimated median estimated dimension is 87.5, but it seems that roughly up to dimension 200, relevant information might be contained.

As explained in the main text, the encoding used by the rbf-kernel is not fit for this example. The four aminoacids A, C, G, and T have just been mapped to the numbers 1–4. We re-encode the object features by mapping A, C, G, and T to the four vectors $(1,0,0,0)$, $(0,1,0,0)$, and so on. The resulting kernel PCA coefficients are plotted in Figure 9(b). The encoding has obviously resulted in a large improvement, as the dimension is much smaller now, while the amount of noise has also been reduced.

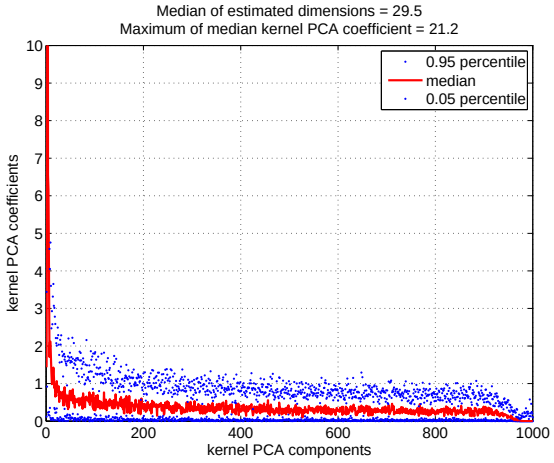
Finally, we consider using a weighted-degree-kernel (Sonnenburg et al., 2005). The resulting kernel PCA coefficients are plotted in Figure 9(c). While the estimated dimension is larger than



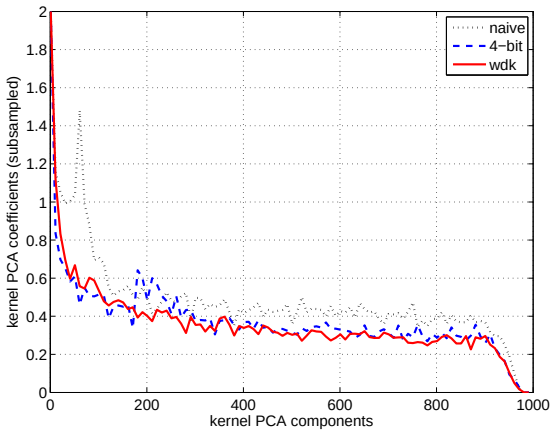
(a) Rbf-kernel on the original data.



(b) Encoding the aminoacids using four bits.



(c) Using a weighted-degree kernel.



(d) The mean kernel PCA coefficients of all three kernels compared (coefficients clipped to the interval from 0 to 2).

Figure 9: Figures (a)-(c) show 0.05, 0.5, and 0.95 percentiles of the kernel PCA coefficients over the 20 resamples of the *splice* data set using the indicated kernels. Coefficients have been truncated to the range $[0, 10]$ for better visibility. Figure (d) plots all three medians for comparison (subsampled by combining ten consecutive points into their mean for better visibility). Coefficients are sorted by decreasing corresponding eigenvalue.

in the previous case, the amount of noise was dramatically reduced, which is also reflected in the classification results shown in Table 5.

In summary, using the estimates here, one can get a much more fine-grained assessment of how well a kernel is adapted to the data. Figure 9(d) compares the mean kernel PCA coefficients over the resamples for the three kernels. Initially, the splice data set appears to be rather high-dimensional, indicating that more data would be needed. Incorporating domain knowledge in the encoding and finally switching to a special-purpose kernel shows that the true dimensionality of the data is in fact

smaller, and that the noise level, which was initially quite high, could also be lowered significantly. Using the weighted-degree-kernel the data quality and the amount of data seem to be suited for predicting with high accuracy.

Appendix D. A Sin-Theta-Theorem

The following theorem is a special case of Theorem 6.2 in the book by Davis and Kahan (1970).

Theorem 2 *Let* $\mathbf{A}$ *be a symmetric* $n \times n$*-matrix with eigendecomposition* $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{U}^\top$*. Let* $\mathbf{U}$ *and* $\mathbf{L}$ *be partitioned as follows:*

$$\mathbf{U} = [\mathbf{U}_1\, \mathbf{U}_2], \qquad \mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & 0 \\ 0 & \mathbf{L}_2 \end{bmatrix},$$

where $\mathbf{U}_1$ *is an* $n \times k$*-matrix,* $\mathbf{L}_1$ *is a* $k \times k$*-matrix,* $\mathbf{U}_2$ *is an* $n \times n-k$*-matrix, and* $\mathbf{L}_2$ *is an* $n-k \times n-k$*-matrix. Furthermore, let* $\mathbf{E}$ *be another symmetric* $n \times n$*-matrix, and* $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{E}$*. Let* $\tilde{l}$ *be an eigenvalue of* $\tilde{\mathbf{A}}$ *and* $\tilde{x}$ *an associated unit-length eigenvector. Then,*

$$\|\mathbf{U}_2^\top \tilde{x}\| \leq \frac{\|\mathbf{E}\|}{\min\limits_{n-k \leq i \leq n} |\tilde{l} - l_i|}.$$

The proof of this theorem can also be found in the thesis of Braun (2005), Lemma 4.50, p. 70.

References

Gilles Blanchard, Olivier Bousquet, and Laurent Zwald. Statistical properties of kernel principal component analysis. *Machine Learning*, 66(2–3):259–294, 2007.

Mikio L. Braun. Accurate error bounds for the eigenvalues of the kernel matrix. *Journal of Machine Learning Research*, 7:2303–2328, Nov 2006.

Mikio L. Braun. *Spectral Properties of the Kernel Matrix and Their Application to Kernel Methods in Machine Learning*. PhD thesis, University of Bonn, 2005. Available electronically at http://hss.ulb.uni-bonn.de/diss_online/math_nat_fak/2005/braun_mikio.

Chris J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

Chandler Davis and William M. Kahan. The rotation of eigenvectors by a perturbation, iii. *SIAM Journal of Numerical Analysis*, 7:1–46, 1970.

Theodoros Evgeniou and Massimiliano Pontil. On the V_γ dimension for regression in reproducing kernel hilbert spaces. In *Proceedings of Algorithmic Learning Theory*, 1999.

Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 160(620–630), 1957.

Vladimir Koltchinskii and Evariste Giné. Random matrix approximation of spectra of integral operators. *Bernoulli*, 6(1):113–167, 2000.

Vladimir I. Koltchinskii. Asymptotics of spectral projections of some random matrices approximating integral operators. *Progress in Probability*, 43:191–227, 1998.

Sebastian Mika. *Kernel Fisher Discriminants*. PhD thesis, Technische Universität Berlin, December 2002.

Klaus-Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transaction on Neural Networks*, 12(2): 181–201, May 2001.

Gunnar Rätsch, Takashi Onoda, and Klaus-Robert Müller. Soft margins for AdaBoost. *Machine Learning*, 42(3):287–320, March 2001.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2002.

Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

Bernhard Schölkopf, Sebastian Mika, Christopher J. C. Burges, Philipp Knirsch, Klaus-Robert Müller, Gunnar Rätsch, and Alex J. Smola. Input space vs. feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, and Martin Anthony. Structural risk minimization over date-dependent hierarchies. *IEEE Transactions on Information Theory*, 44(5): 1926–1940, 1998.

John Shawe-Taylor, Christopher K. I. Williams, Nello Christianini, and Jaz Kandola. On the eigenspectrum of the Gram matrix and the generalization error of kernel-PCA. *IEEE Transactions on Information Theory*, 51(7):2510–2522, July 2005.

Alex J. Smola, Bernhard Schölkopf, and Klaus-Robert Müller. The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649, 1998.

Sören Sonnenburg, Gunnar Rätsch, and Bernhard Schölkopf. Large scale genomic sequence SVM classifiers. In *Proceedings of the 22nd International Machine Learning Conference*, pages 848–855. ACM Press, 2005.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Vladimir Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Régis Vert, Laurent Zwald, Gilles Blanchard, and Pascal Massart. Kernel projection machine: a new tool for pattern recognition. In *Advances in Neural Information Processing Systems (NIPS 2004)*, pages 1649–1656. 2005, 2005.

Ulrike von Luxburg. *Statistical Learning with Similarity and Dissimilarity Functions*. PhD thesis, Technische Universität Berlin, 2004.

Grace Wahba. *Spline Models For Observational Data*. Society for Industrial and Applied Mathematics, 1990.

Robert C. Williamson, Alex J. Smola, and Bernhard Schölkopf. Generalization bounds for regularization networks and support vector machines via entropy numbers of compact operators. *IEEE Transaction on Information Theory*, 47(6):2516–2532, 2001.

Tong Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17:2077–2098, 2005.

Laurent Zwald and Gilles Blanchard. On the convergence of eigenspaces in kernel principal components analysis. In *Advances in Neural Information Processing Systems (NIPS 2005)*, volume 18, 2006.

Journal of Machine Learning Research 9 (2008) 1909-1939 Submitted 12/07; Revised 5/08; Published 8/08

Manifold Learning: The Price of Normalization

Yair Goldberg YAIRGO@CC.HUJI.AC.IL
Department of Statistics
The Hebrew University
91905 Jerusalem, Israel

Alon Zakai ALONZAKA@POB.HUJI.AC.IL
Interdisciplinary Center for Neural Computation
The Hebrew University
91905 Jerusalem, Israel

Dan Kushnir DAN.KUSHNIR@WEIZMANN.AC.IL
Department of Computer Science and Applied Mathematics
The Weizmann Institute of Science
76100 Rehovot, Israel

Ya'acov Ritov YAACOV.RITOV@HUJI.AC.IL
Department of Statistics
The Hebrew University
91905 Jerusalem, Israel

Editor: Sam Roweis

Abstract

We analyze the performance of a class of manifold-learning algorithms that find their output by minimizing a quadratic form under some normalization constraints. This class consists of Locally Linear Embedding (LLE), Laplacian Eigenmap, Local Tangent Space Alignment (LTSA), Hessian Eigenmaps (HLLE), and Diffusion maps. We present and prove conditions on the manifold that are necessary for the success of the algorithms. Both the finite sample case and the limit case are analyzed. We show that there are simple manifolds in which the necessary conditions are violated, and hence the algorithms cannot recover the underlying manifolds. Finally, we present numerical results that demonstrate our claims.

Keywords: dimensionality reduction, manifold learning, Laplacian eigenmap, diffusion maps, locally linear embedding, local tangent space alignment, Hessian eigenmap

1. Introduction

Many seemingly complex systems described by high-dimensional data sets are in fact governed by a surprisingly low number of parameters. Revealing the low-dimensional representation of such high-dimensional data sets not only leads to a more compact description of the data, but also enhances our understanding of the system. Dimension-reducing algorithms attempt to simplify the system's representation without losing significant structural information. Various dimension-reduction algorithms were developed recently to perform embeddings for manifold-based data sets. These include the following algorithms: Locally Linear Embedding (LLE, Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000), Laplacian Eigenmaps (LEM, Belkin and Niyogi, 2003), Local Tangent Space Alignment (LTSA, Zhang and Zha, 2004), Hessian Eigenmap (HLLE, Donoho and Grimes,

2004), Semi-definite Embedding (SDE, Weinberger and Saul, 2006) and Diffusion Maps (DFM, Coifman and Lafon, 2006).

These manifold-learning algorithms compute an embedding for some given input. It is assumed that this input lies on a low-dimensional manifold, embedded in some high-dimensional space. Here a manifold is defined as a topological space that is locally equivalent to a Euclidean space. It is further assumed that the manifold is the image of a low-dimensional domain. In particular, the input points are the image of a sample taken from the domain. The goal of the manifold-learning algorithms is to recover the original domain structure, up to some scaling and rotation. The non-linearity of these algorithms allows them to reveal the domain structure even when the manifold is not linearly embedded.

The central question that arises when considering the output of a manifold-learning algorithm is, whether the algorithm reveals the underlying low-dimensional structure of the manifold. The answer to this question is not simple. First, one should define what "revealing the underlying lower-dimensional description of the manifold" actually means. Ideally, one could measure the degree of similarity between the output and the original sample. However, the original low-dimensional data representation is usually unknown. Nevertheless, if the low-dimensional structure of the data is known in advance, one would expect it to be approximated by the dimension-reducing algorithm, at least up to some rotation, translation, and global scaling factor. Furthermore, it would be reasonable to expect the algorithm to succeed in recovering the original sample's structure asymptotically, namely, when the number of input points tends to infinity. Finally, one would hope that the algorithm would be robust in the presence of noise.

Previous papers have addressed the central question posed earlier. Zhang and Zha (2004) presented some bounds on the local-neighborhoods' error-estimation for LTSA. However, their analysis says nothing about the global embedding. Huo and Smith (2006) proved that, asymptotically, LTSA recovers the original sample up to an affine transformation. They assume in their analysis that the level of noise tends to zero when the number of input points tends to infinity. Bernstein et al. (2000.) proved that, asymptotically, the embedding given by the Isomap algorithm (Tenenbaum et al., 2000) recovers the geodesic distances between points on the manifold.

In this paper we develop theoretical results regarding the performance of a class of manifold-learning algorithms, which includes the following five algorithms: Locally Linear Embedding (LLE), Laplacian Eigenmap (LEM), Local Tangent Space Alignment (LTSA), Hessian Eigenmaps (HLLE), and Diffusion maps (DFM).

We refer to this class of algorithms as the normalized-output algorithms. The normalized-output algorithms share a common scheme for recovering the domain structure of the input data set. This scheme is constructed in three steps. In the first step, the local neighborhood of each point is found. In the second step, a description of these neighborhoods is computed. In the third step, a low-dimensional output is computed by solving some convex optimization problem under some normalization constraints. A detailed description of the algorithms is given in Section 2.

In Section 3 we discuss informally the criteria for determining the success of manifold-learning algorithms. We show that one should not expect the normalized-output algorithms to recover geodesic distances or local structures. A more reasonable criterion for success is a high degree of similarity between the output of the algorithms and the original sample, up to some affine transformation; the definition of similarity will be discussed later. We demonstrate that under certain circumstances, this high degree of similarity does not occur. In Section 4 we find necessary conditions for the successful performance of LEM and DFM on the two-dimensional grid. This section serves

as an explanatory introduction to the more general analysis that appears in Section 5. Some of the ideas that form the basis of the analysis in Section 4 were discussed independently by both Gerber et al. (2007) and ourselves (Goldberg et al., 2007). Section 5 finds necessary conditions for the successful performance of all the normalized-output algorithms on general two-dimensional manifolds. It should be noted that the necessary conditions are hard to verify in practice. However, they serve as an analytic tool to prove that there are general classes of manifolds on which the normalized-output algorithms fail. Moreover, the numerical examples in this section show that the class of manifolds on which the normalized-output algorithms fail is wide and includes non-isometrically manifolds and real-world data. In Section 6 we discuss the performance of the algorithms in the asymptotic case. Concluding remarks appear in Section 7. The detailed proofs appear in the Appendix.

Our paper has two main results. First, we give well-defined necessary conditions for the successful performance of the normalized-output algorithms. Second, we show that there exist simple manifolds that do not fulfill the necessary conditions for the success of the algorithms. For these manifolds, the normalized-output algorithms fail to generate output that recovers the structure of the original sample. We show that these results hold asymptotically for LEM and DFM. Moreover, when noise, even of small variance, is introduced, LLE, LTSA, and HLLE will fail asymptotically on some manifolds. Throughout the paper, we present numerical results that demonstrate our claims.

2. Description of Output-normalized Algorithms

In this section we describe in short the normalized-output algorithms. The presentation of these algorithms is not in the form presented by the respective authors. The form used in this paper emphasizes the similarities between the algorithms and is better-suited for further derivations. In Appendix A.1 we show the equivalence of our representation of the algorithms and the representations that appear in the original papers.

Let $X = [x_1, \ldots, x_N]'$, $x_i \in \mathbb{R}^{\mathcal{D}}$ be the input data where $\mathcal{D}$ is the dimension of the ambient space and N is the size of the sample. The normalized-output algorithms attempt to recover the underlying structure of the input data X in three steps.

In the first step, the normalized-output algorithms assign neighbors to each input point x_i based on the Euclidean distances in the high-dimensional space.[1] This can be done, for example, by choosing all the input points in an r-ball around x_i or alternatively by choosing x_i's K-nearest-neighbors. The neighborhood of x_i is given by the matrix $X_i = [x_i, x_{i,1}, \ldots, x_{i,K}]'$ where $x_{i,j} : j = 1, \ldots, K$ are the neighbors of x_i. Note that $K = K(i)$ can be a function of i, the index of the neighborhood, yet we omit this index to simplify the notation. For each neighborhood, we define the radius of the neighborhood as

$$r(i) = \max_{j,k \in \{0,\ldots,K\}} \left\| x_{i,j} - x_{i,k} \right\|$$

where we define $x_{i,0} = x_i$. Finally, we assume throughout this paper that the neighborhood graph is connected.

In the second step, the normalized-output algorithms compute a description of the local neighborhoods that were found in the previous step. The description of the i-th neighborhood is given by some weight matrix W_i. The matrices W_i for the different algorithms are presented.

1. The neighborhoods are not mentioned explicitly by Coifman and Lafon (2006). However, since a sparse optimization problem is considered, it is assumed implicitly that neighborhoods are defined (see Sec. 2.7 therein).

- LEM and DFM: W_i is a $K \times (K+1)$ matrix,

$$W_i = \begin{pmatrix} w_{i,1}^{1/2} & -w_{i,1}^{1/2} & 0 & \cdots & 0 \\ w_{i,2}^{1/2} & 0 & -w_{i,2}^{1/2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ w_{i,K}^{1/2} & 0 & \cdots & 0 & -w_{i,K}^{1/2} \end{pmatrix}.$$

  For LEM $w_{i,j} = 1$ is a natural choice, yet it is also possible to define the weights as $\tilde{w}_{i,j} = e^{-\left\|x_i - x_{i,j}\right\|^2/\varepsilon}$, where ε is the width parameter of the kernel. For the case of DFM,

$$w_{i,j} = \frac{k_\varepsilon(x_i, x_{i,j})}{q_\varepsilon(x_i)^\alpha q_\varepsilon(x_{i,j})^\alpha}, \tag{1}$$

  where k_ε is some rotation-invariant kernel, $q_\varepsilon(x_i) = \sum_j k_\varepsilon(x_i, x_{i,j})$ and ε is again a width parameter. We will use $\alpha = 1$ in the normalization of the diffusion kernel, yet other values of α can be considered (see details in Coifman and Lafon, 2006). For both LEM and DFM, we define the matrix D to be a diagonal matrix where $d_{ii} = \sum_j w_{i,j}$.

- LLE: W_i is a $1 \times (K+1)$ matrix,

$$W_i = \begin{pmatrix} 1 & -w_{i,1} & \cdots & -w_{i,K} \end{pmatrix}.$$

  The weights $w_{i,j}$ are chosen so that x_i can be best linearly reconstructed from its neighbors. The weights minimize the reconstruction error function

$$\Delta^i(w_{i,1}, \ldots, w_{i,K}) = \|x_i - \sum_j w_{i,j} x_{i,j}\|^2$$

  under the constraint $\sum_j w_{i,j} = 1$. In the case where there is more than one solution that minimizes Δ^i, regularization is applied to force a unique solution (for details, see Saul and Roweis, 2003).

- LTSA: W_i is a $(K+1) \times (K+1)$ matrix,

$$W_i = (I - P_i P_i')H.$$

  Let $U_i L_i V_i'$ be the SVD of $X_i - \mathbf{1}\bar{x_i}'$ where $\bar{x_i}$ is the sample mean of X_i and $\mathbf{1}$ is a vector of ones (for details about SVD, see, for example, Golub and Loan, 1983). Let $P_i = [u_{(1)}, \ldots, u_{(d)}]$ be the matrix that holds the first d columns of U_i where d is the output dimension. The matrix $H = I - \frac{1}{K}\mathbf{1}\mathbf{1}'$ is the centering matrix. See also Huo and Smith (2006) regarding this representation of the algorithm.

- HLLE: W_i is a $d(d+1)/2 \times (K+1)$ matrix,

$$W_i = (\mathbf{0}, H^i)$$

  where $\mathbf{0}$ is a vector of zeros and H^i is the $\frac{d(d+1)}{2} \times K$ Hessian estimator.
  The estimator can be calculated as follows. Let $U_i L_i V_i'$ be the SVD of $X_i - \mathbf{1}\bar{x_i}'$. Let

$$M_i = [\mathbf{1}, U_i^{(1)}, \ldots, U_i^{(d)}, \mathrm{diag}(U_i^{(1)} U_i^{(1)\prime}), \mathrm{diag}(U_i^{(1)} U_i^{(2)\prime}), \ldots, \mathrm{diag}(U_i^{(d)} U_i^{(d)\prime})],$$

where the operator diag returns a column vector formed from the diagonal elements of the matrix. Let $\widetilde{M}_i$ be the result of the Gram-Schmidt orthonormalization on M_i. Then H^i is defined as the transpose of the last $d(d+1)/2$ columns of $\widetilde{M}_i$.

The third step of the normalized-output algorithms is to find a set of points $Y = [y_1, \ldots, y_N]', y_i \in \mathbb{R}^d$ where $d \leq \mathcal{D}$ is the dimension of the manifold. Y is found by minimizing a convex function under some normalization constraints, as follows. Let Y be any $N \times d$ matrix. We define the i-th neighborhood matrix $Y_i = [y_i, y_{i,1}, \ldots, y_{i,K}]'$ using the same pairs of indices i, j as in X_i. The cost function for all of the normalized-output algorithms is given by

$$\Phi(Y) = \sum_{i=1}^{N} \phi(Y_i) = \sum_{i=1}^{N} \|W_i Y_i\|_F^2 \,, \tag{2}$$

under the normalization constraints

$$\begin{cases} Y'DY = I \\ Y'D\mathbf{1} = \mathbf{0} \end{cases} \text{for LEM and DFM,} \quad \begin{cases} \mathrm{Cov}(Y) = I \\ Y'\mathbf{1} = \mathbf{0} \end{cases} \text{for LLE, LTSA and HLLE,} \tag{3}$$

where $\| \|_F$ stands for the Frobenius norm, and W_i is algorithm-dependent.

Define the output matrix Y to be the matrix that achieves the minimum of Φ under the normalization constraints of Eq. 3 (Y is defined up to rotation). Then we have the following: the embeddings of LEM and LLE are given by the according output matrices Y; the embeddings of LTSA and HLLE are given by the according output matrices $\frac{1}{\sqrt{N}}Y$; and the embedding of DFM is given by a linear transformation of Y as discussed in Appendix A.1. The discussion of the algorithms' output in this paper holds for any affine transformation of the output (see Section 3). Thus, without loss of generality, we prefer to discuss the output matrix Y directly, rather than the different embeddings. This allows a unified framework for all five normalized-output algorithms.

3. Embedding Quality

In this section we discuss possible definitions of "successful performance" of manifold-learning algorithms. To open our discussion, we present a numerical example. We chose to work with LTSA rather arbitrarily. Similar results can be obtained using the other algorithms.

The example we consider is a uniform sample from a two-dimensional strip, shown in Fig. 1A. Note that in this example, $\mathcal{D} = d$; that is, the input data is identical to the original data. Fig. 1B presents the output of LTSA on the input in Fig. 1A. The most obvious difference between input and output is that while the input is a strip, the output is roughly square. While this may seem to be of no importance, note that it means that the algorithm, like all the normalized-output algorithms, does not preserve geodesic distances even up to a scaling factor. By definition, the geodesic distance between two points on a manifold is the length of the shortest path on the manifold between the two points. Preservation of geodesic distances is particularly relevant when the manifold is isometrically embedded. In this case, assuming the domain is convex, the geodesic distance between any two points on the manifold is equal to the Euclidean distance between the corresponding domain points. Geodesic distances are conserved, for example, by the Isomap algorithm (Tenenbaum et al., 2000).

Figs. 1E and 1F present closeups of Figs. 1A and 1B, respectively. Here, a less obvious phenomenon is revealed: the structure of the local neighborhood is not preserved by LTSA. By local structure we refer to the angles and distances (at least up to a scale) between all points within each

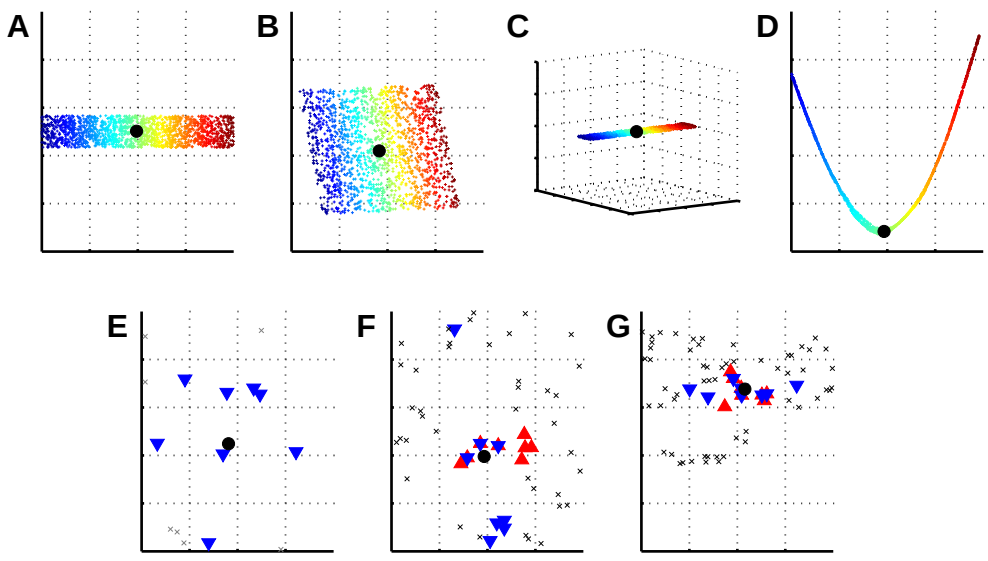


Figure 1: The output of LTSA (B) for the (two-dimensional) input shown in (A), where the input is a uniform sample from the strip $[0,1] \times [0,6]$. Ideally one would expect the two to be identical. The normalization constraint shortens the horizontal distances and lengthens the vertical distances, leading to the distortion of geodesic distances. (E) and (F) focus on the points shown in black in (A) and (B), respectively. The (blue) triangles pointing downwards in (E) and (F) are the 8-nearest-neighborhood of the point denoted by the full black circle. The (red) triangles pointing upwards in (F) indicate the neighborhood computed for the corresponding point (full black circle) in the output space. Note that less than half of the original neighbors of the point remain neighbors in the output space. The input (A) with the addition of Gaussian noise normal to the manifold and of variance 10^{-4} is shown in (C). The output of LTSA for the noisy input is shown in (D). (G) shows a closeup of the neighborhood of the point indicated by the black circle in (D).

local neighborhood. Mappings that preserve local structures up to a scale are called conformal mappings (see for example de Silva and Tenenbaum, 2003; Sha and Saul, 2005). In addition to the distortion of angles and distances, the K-nearest-neighbors of a given point on the manifold do not necessarily correspond to the K-nearest-neighbors of the respective output point, as shown in Figs. 1E and 1F. Accordingly, we conclude that the original structure of the local neighborhoods is not necessarily preserved by the normalized-output algorithms.

The above discussion highlights the fact that one cannot expect the normalized-output algorithms to preserve geodesic distances or local neighborhood structure. However, it seems reasonable to demand that the output of the normalized-output algorithms resemble an affine transformation of the original sample. In fact, the output presented in Fig. 1B is an affine transformation of the input, which is the original sample, presented in Fig. 1A. A formal similarity criterion based on affine transformations is given by Huo and Smith (2006). In the following, we will claim that a

normalized-output algorithm succeeds (or fails) based on the existence (or lack thereof) of resemblance between the output and the original sample, up to an affine transformation.

Fig. 1D presents the output of LTSA on a noisy version of the input, shown in Fig. 1C. In this case, the algorithm prefers an output that is roughly a one-dimensional curve embedded in $\mathbb{R}^2$. While this result may seem incidental, the results of all the other normalized-output algorithms for this example are essentially the same.

Using the affine transformation criterion, we can state that LTSA succeeds in recovering the underlying structure of the strip shown in Fig. 1A. However, in the case of the noisy strip shown in Fig. 1C, LTSA fails to recover the structure of the input. We note that all the other normalized-output algorithms perform similarly.

For practical purposes, we will now generalize the definition of failure of the normalized-output algorithms. This definition is more useful when it is necessary to decide whether an algorithm has failed, without actually computing the output. This is useful, for example, when considering the outputs of an algorithm for a class of manifolds.

We now present the generalized definition of failure of the algorithms. Let $X = X_{N\times d}$ be the original sample. Assume that the input is given by $\psi(X) \subset \mathbb{R}^{\mathcal{D}}$, where $\psi : \mathbb{R}^d \to \mathbb{R}^{\mathcal{D}}$ is some smooth function, and $\mathcal{D} \geq d$ is the dimension of the input. Let $Y = Y_{N\times d}$ be an affine transformation of the original sample X, such that the normalization constraints of Eq. 3 hold. Note that Y is algorithm-dependent, and that for each algorithm, Y is unique up to rotation and translation. When the algorithm succeeds it is expected that the output will be similar to a normalized version of X, namely to Y. Let $Z = Z_{N\times d}$ be any matrix that satisfies the same normalization constraints. We say that the algorithm has failed if $\Phi(Y) > \Phi(Z)$, and Z is substantially different from Y, and hence also from X. In other words, we say that the algorithm has failed when a substantially different embedding Z has a lower cost than the most appropriate embedding Y. A precise definition of "substantially different" is not necessary for the purposes of this paper. It is enough to consider Z substantially different from Y when Z is of lower dimension than Y, as in Fig. 1D.

We emphasize that the matrix Z is not necessarily similar to the output of the algorithm in question. It is a mathematical construction that shows when the output of the algorithm is not likely to be similar to Y, the normalized version of the true manifold structure. The following lemma shows that if $\Phi(Y) > \Phi(Z)$, the inequality is also true for a small perturbation of Y. Hence, it is not likely that an output that resembles Y will occur when $\Phi(Y) > \Phi(Z)$ and Z is substantially different from Y.

Lemma 3.1 *Let Y be an $N \times d$ matrix. Let $\widetilde{Y} = Y + \varepsilon E$ be a perturbation of Y, where E is an $N \times d$ matrix such that $\|E\|_F = 1$ and where $\varepsilon > 0$. Let S be the maximum number of neighborhoods to which a single input point belongs. Then for LLE with positive weights $w_{i,j}$, LEM, DFM, LTSA, and HLLE, we have*

$$\Phi(\widetilde{Y}) > (1-\varepsilon)\Phi(Y) - \varepsilon C_a S,$$

where C_a is a constant that depends on the algorithm.

The use of positive weights in LLE is discussed in Saul and Roweis (2003, Section 5); a similar result for LLE with general weights can be obtained if one allows a bound on the values of $w_{i,j}$. The proof of Lemma 3.1 is given in Appendix A.2.

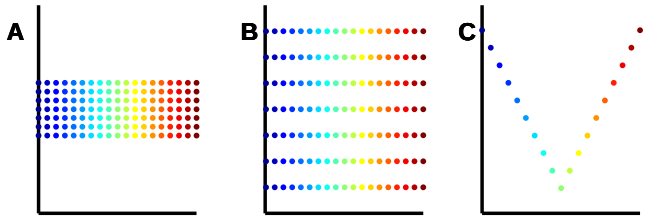


Figure 2: (A) The input grid. (B) Embedding Y, the normalized grid. (C) Embedding Z, a curve that satisfies $\mathrm{Cov}(Z) = I$.

4. Analysis of the Two-Dimensional Grid

In this section we analyze the performance of LEM on the two-dimensional grid. In particular, we argue that LEM cannot recover the structure of a two-dimensional grid in the case where the aspect ratio of the grid is greater than 2. Instead, LEM prefers a one-dimensional curve in $\mathbb{R}^2$. Implications also follow for DFM, as explained in Section 4.3, followed by a discussion of the other normalized-output algorithms. Finally, we present empirical results that demonstrate our claims.

In Section 5 we prove a more general statement regarding any two-dimensional manifold. Necessary conditions for successful performance of the normalized-output algorithms on such manifolds are presented. However, the analysis in this section is important in itself for two reasons. First, the conditions for the success of LEM on the two-dimensional grid are more limiting. Second, the analysis is simpler and points out the reasons for the failure of all the normalized-output algorithms when the necessary conditions do not hold.

4.1 Possible Embeddings of a Two-Dimensional Grid

We consider the input data set X to be the two-dimensional grid $[-m,\ldots,m]\times[-q,\ldots,q]$, where $m \geq q$. We denote $x_{ij} = (i, j)$. For convenience, we regard $X = (X^{(1)}, X^{(2)})$ as an $N \times 2$ matrix, where $N = (2m+1)(2q+1)$ is the number of points in the grid. Note that in this specific case, the original sample and the input are the same.

In the following we present two different embeddings, Y and Z. Embedding Y is the grid itself, normalized so that $\mathrm{Cov}(Y) = I$. Embedding Z collapses each column to a point and positions the resulting points in the two-dimensional plane in a way that satisfies the constraint $\mathrm{Cov}(Z) = I$ (see Fig. 2 for both). The embedding Z is a curve in $\mathbb{R}^2$ and clearly does not preserve the original structure of the grid.

We first define the embeddings more formally. We start by defining $\widehat{Y} = X(X'DX)^{-1/2}$. Note that this is the only linear transformation of X (up to rotation) that satisfies the conditions $\widehat{Y}'D\mathbf{1} = \mathbf{0}$ and $\widehat{Y}'D\widehat{Y} = I$, which are the normalization constraints for LEM (see Eq. 3). However, the embedding $\widehat{Y}$ depends on the matrix D, which in turn depends on the choice of neighborhoods. Recall that the matrix D is a diagonal matrix, where d_{ii} equals the number of neighbors of the i-th point. Choose r to be the radius of the neighborhoods. Then, for all inner points x_{ij}, the number of neighbors $K(i, j)$ is a constant, which we denote as K. We shall call all points with less than K neighbors *boundary*

points. Note that the definition of boundary points depends on the choice of r. For inner points of the grid we have $d_{ii} \equiv K$. Thus, when $K \ll N$ we have $X'DX \approx KX'X$.

We define $Y = X\mathrm{Cov}(X)^{-1/2}$. Note that $Y'\mathbf{1} = 0$, $\mathrm{Cov}(Y) = I$ and for $K \ll N$, $Y \approx \sqrt{KN}\widehat{Y}$. In this section we analyze the embedding Y instead of $\widehat{Y}$, thereby avoiding the dependence on the matrix D and hence simplifying the notation. This simplification does not significantly change the problem and does not affect the results we present. Similar results are obtained in the next section for general two-dimensional manifolds, using the exact normalization constraints (see Section 5.2).

Note that Y can be described as the set of points $[-m/\sigma,\ldots,m/\sigma]\times[-q/\tau,\ldots,q/\tau]$, where $y_{ij} = (i/\sigma, j/\tau)$. The constants $\sigma^2 = \mathrm{Var}(X^{(1)})$ and $\tau^2 = \mathrm{Var}(X^{(2)})$ ensure that the normalization constraint $\mathrm{Cov}(Y) = I$ holds. Straightforward computation (see Appendix A.3) shows that

$$\sigma^2 = \frac{(m+1)m}{3};\ \tau^2 = \frac{(q+1)q}{3}. \tag{4}$$

The definition of the embedding Z is as follows:

$$z_{ij} = \begin{cases} \left(\frac{i}{\sigma}, \frac{-2i}{\rho} - \overline{z}^{(2)}\right) & i \le 0 \\ \\ \left(\frac{i}{\sigma}, \frac{2i}{\rho} - \overline{z}^{(2)}\right) & i \ge 0 \end{cases},$$

where $\overline{z}^{(2)} = \frac{(2q+1)2}{N\rho}\sum_{i=1}^{m}(2i)$ ensures that $Z'\mathbf{1} = \mathbf{0}$, and σ (the same σ as before; see below) and ρ are chosen so that sample variance of $Z^{(1)}$ and $Z^{(2)}$ is equal to one. The symmetry of $Z^{(1)}$ about the origin implies that $\mathrm{Cov}(Z^{(1)}, Z^{(2)}) = 0$, hence the normalization constraint $\mathrm{Cov}(Z) = I$ holds. σ is as defined in Eq. 4, since $Z^{(1)} = Y^{(1)}$ (with both defined similarly to $X^{(1)}$). Finally, note that the definition of z_{ij} does not depend on j.

4.2 Main Result for LEM on the Two-Dimensional Grid

We estimate $\Phi(Y)$ by $N\phi(Y_{ij})$ (see Eq. 2), where y_{ij} is an inner point of the grid and Y_{ij} is the neighborhood of y_{ij}; likewise, we estimate $\Phi(Z)$ by $N\phi(Z_{ij})$ for an inner point z_{ij}. For all inner points, the value of $\phi(Y_{ij})$ is equal to some value ϕ. For boundary points, $\phi(Y_{ij})$ is bounded by ϕ multiplied by some constant that depends only on the number of neighbors. Hence, for large m and q, the difference between $\Phi(Y)$ and $N\phi(Y_{ij})$ is negligible.

The main result of this section states:

Theorem 4.1 *Let y_{ij} be an inner point and let the ratio $\frac{m}{q}$ be greater than* 2. *Then*

$$\phi(Y_{ij}) > \phi(Z_{ij})$$

for neighborhood-radius r that satisfies $1 \le r \le 3$, or similarly, for K-nearest neighborhoods where $K = 4, 8, 12$.

This indicates that for aspect ratios $\frac{m}{q}$ that are greater than 2 and above, mapping Z, which is essentially one-dimensional, is preferred to Y, which is a linear transformation of the grid. The case of general r-ball neighborhoods is discussed in Appendix A.4 and indicates that similar results should be expected.

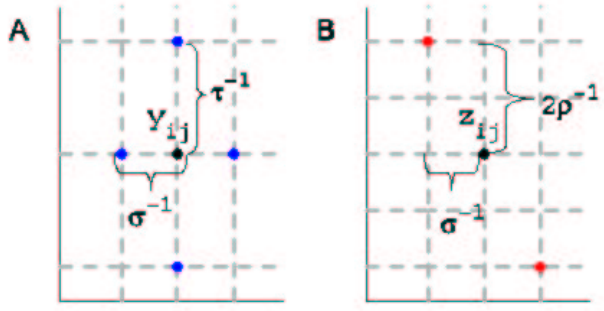


Figure 3: (A) The normalized grid at an inner point y_{ij}. The 4-nearest-neighbors of y_{ij} are marked in blue. Note that the neighbors from the left and from the right are at a distance of $1/\sigma$, while the neighbors from above and below are at a distance of $1/\tau$. The value of $\phi(Y_{ij})$ is equal to the sum of squared distances of y_{ij} to its neighbors. Hence, we obtain that $\phi(Y_{ij}) = 2/\sigma^2 + 2/\tau^2$ when $K = 4$ and $\phi(Y_{ij}) = 2/\sigma^2 + 2/\tau^2 + 4(1/\sigma^2 + 1/\tau^2)$ when $K = 8$. (B) The curve embedding at an inner point z_{ij}. The neighbors of z_{ij} from the left and from the right are marked in red. The neighbors from above and below are embedded to the same point as z_{ij}. Note that the squared distance between z_{ij} and $z_{(i\pm1)j}$ equals $1/\sigma^2 + 4/\rho^2$. Hence, $\phi(Z_{ij}) = 2(1/\sigma^2 + 4/\rho^2)$ when $K = 4$, and $\phi(Z_{ij}) = 6(1/\sigma^2 + 4/\rho^2)$ when $K = 8$.

The proof of the theorem is as follows. It can be shown analytically (see Fig. 3) that

$$\phi(Y_{ij}) = F(K)\left(\frac{1}{\sigma^2} + \frac{1}{\tau^2}\right), \tag{5}$$

where

$$F(4) = 2;\ F(8) = 6;\ F(12) = 14.$$

For higher K, $F(K)$ can be approximated for any r-ball neighborhood of y_{ij} (see Appendix A.4).

It can be shown (see Fig. 3) that

$$\phi(Z_{ij}) = \widetilde{F}(K)\left(\frac{1}{\sigma^2} + \frac{4}{\rho^2}\right), \tag{6}$$

where $\widetilde{F}(K) = F(K)$ for $K = 4, 8, 12$. For higher K, it can be shown (see Appendix A.4) that $\widetilde{F}(K) \approx F(K)$ for any r-ball neighborhood.

A careful computation (see Appendix A.5) shows that

$$\rho > \sigma, \tag{7}$$

and therefore

$$\phi(Z_{ij}) < \frac{5F(K)}{\sigma^2}. \tag{8}$$

Assume that $\frac{m}{q} > 2$. Since both m and q are integers, we have that $m+1 \geq 2(q+1)$. Hence, using Eq. 4 we have

$$\sigma^2 = \frac{m(m+1)}{3} > \frac{4q(q+1)}{3} = 4\tau^2 .$$

Combining this result with Eqs. 5 and 8 we have

$$\frac{m}{q} > 2 \Rightarrow \phi(Y_{ij}) > \phi(Z_{ij})$$

which proves Theorem 4.1.

4.3 Implications to Other Algorithms

We start with implications regarding DFM. There are two main differences between LEM and DFM. The first difference is the choice of the kernel. LEM chooses $w_{i,j} = 1$, which can be referred to as the "window" kernel (a Gaussian weight function was also considered by Belkin and Niyogi, 2003). DFM allows a more general rotation-invariant kernel, which includes the "window" kernel of LEM. The second difference is that DFM renormalizes the weights $k_\varepsilon(x_i, x_{i,j})$ (see Eq. 1). However, for all the inner points of the grid with neighbors that are also inner points, the renormalization factor $(q_\varepsilon(x_i)^{-1} q_\varepsilon(x_{i,j})^{-1})$ is a constant. Therefore, if DFM chooses the "window" kernel, it is expected to fail, like LEM. In other words, when DFM using the "window" kernel is applied to a grid with aspect ratio slightly greater than 2 or above, DFM will prefer the embedding Z over the embedding Y (see Fig 2). For a more general choice of kernel, the discussion in Appendix A.4 indicates that a similar failure should occur. This is because the relation between the estimations of $\Phi(Y)$ and $\Phi(Z)$ presented in Eqs. 5 and 6 holds for any rotation-invariant kernel (see Appendix A.4). This observation is also evident in numerical examples, as shown in Figs. 4 and 5.

In the cases of LLE with no regularization, LTSA, and HLLE, it can be shown that $\Phi(Y) \equiv 0$. Indeed, for LTSA and HLLE, the weight matrix W_i projects on a space that is perpendicular to the SVD of the neighborhood X_i, thus $\|W_i X_i\|_F^2 = 0$. Since $Y_i = X_i \mathrm{Cov}(X)^{-1/2}$, we have $\|W_i Y_i\|_F^2 = 0$, and, therefore, $\Phi(Y) \equiv 0$. For the case of LLE with no regularization, when $K \geq 3$, each point can be reconstructed perfectly from its neighbors, and the result follows. Hence, a linear transformation of the original data should be the preferred output. However, the fact that $\Phi(Y) \equiv 0$ relies heavily on the assumption that both the input X and the output Y are of the same dimension (see Theorem 5.1 for manifolds embedded in higher dimensions), which is typically not the case in dimension-reducing applications.

4.4 Numerical Results

For the following numerical results, we used the Matlab implementation written by the respective algorithms' authors as provided by Wittman (retrieved Jan. 2007) (a minor correction was applied to the code of HLLE).

We ran the LEM algorithm on data sets with aspect ratios above and below 2. We present results for both a grid and a uniformly sampled strip. The neighborhoods were chosen using K-nearest neighbors with $K = 4, 8, 16$, and 64. We present the results for $K = 8$; the results for $K = 4, 16$, and 64 are similar. The results for the grid and the random sample are presented in Figs. 4 and 5, respectively.

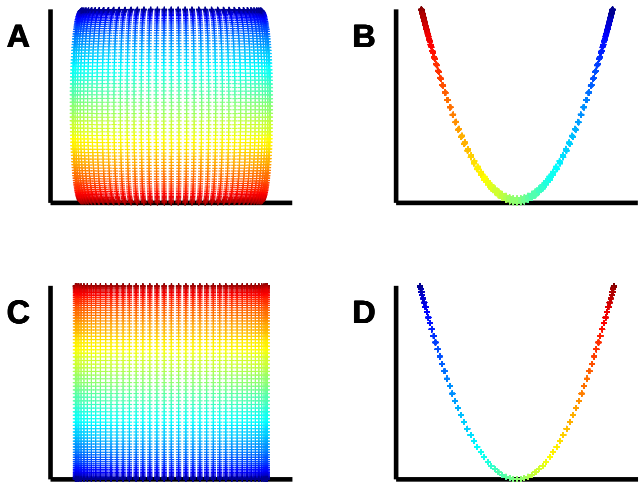


Figure 4: The output of LEM on a grid of dimensions 81×41 is presented in (A). The result of LEM for the grid of dimensions 81×39 is presented in (B). The number of neighbors in both computations is 8. The output for DFM on the same data sets using $\sigma = 2$ appears in (C) and (D), respectively.

We ran the DFM algorithm on the same data sets. We used the normalization constant $\alpha = 1$ and the kernel width $\sigma = 2$; the results for $\sigma = 1, 4$, and 8 are similar. The results for the grid and the random sample are presented in Figures 4 and 5, respectively.

Both examples clearly demonstrate that for aspect ratios sufficiently greater than 2, both LEM and DFM prefer a solution that collapses the input data to a nearly one-dimensional output, normalized in $\mathbb{R}^2$. This is exactly as expected, based on our theoretical arguments.

Finally, we ran LLE, HLLE, and LTSA on the same data sets. In the case of the grid, both LLE and LTSA (roughly) recovered the grid shape for $K = 4, 8, 16$, and 64, while HLLE failed to produce any output due to large memory requirements. In the case of the random sample, both LLE and HLLE succeeded for $K = 16, 64$ but failed for $K = 4, 8$. LTSA succeeded for $K = 8, 16$, and 64 but failed for $K = 4$. The reasons for the failure for lower values of K are not clear, but may be due to roundoff errors. In the case of LLE, the failure may also be related to the use of regularization in LLE's second step.

5. Analysis for General Two-Dimensional Manifolds

The aim of this section is to present necessary conditions for the success of the normalized-output algorithms on general two-dimensional manifolds embedded in high-dimensional space. We show how this result can be further generalized to manifolds of higher dimension. We demonstrate the theoretical results using numerical examples.

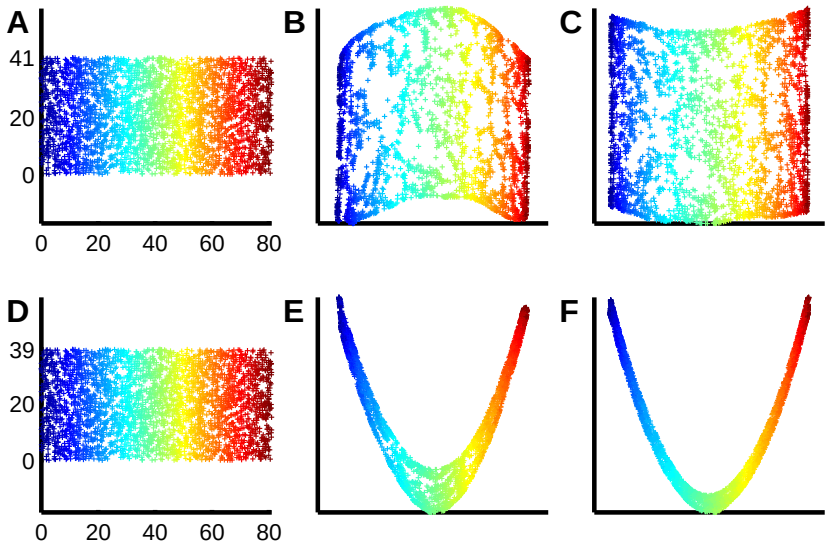


Figure 5: (A) and (D) show the same 3000 points, uniformly-sampled from the unit square, scaled to the areas $[0,81]\times[0,41]$ and $[0,81]\times[0,39]$, respectively. (B) and (E) show the outputs of LEM for inputs (A) and (D), respectively. The number of neighbors in both computations is 8. (C) and (F) show the output for DFM on the same data sets using $\sigma = 2$. Note the sharp change in output structure for extremely similar inputs.

5.1 Two Different Embeddings for a Two-Dimensional Manifold

We start with some definitions. Let $X = [x_1,\ldots,x_N]'$, $x_i \in \mathbb{R}^2$ be the original sample. Without loss of generality, we assume that

$$\bar{x} = \mathbf{0}; \quad \mathrm{Cov}(X) \equiv \Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \tau^2 \end{pmatrix}.$$

As in Section 4, we assume that $\sigma > \tau$. Assume that the input for the normalized-output algorithms is given by $\psi(X) \subset \mathbb{R}^{\mathcal{D}}$ where $\psi : \mathbb{R}^2 \to \mathbb{R}^{\mathcal{D}}$ is a smooth function and $\mathcal{D} \geq 2$ is the dimension of the input. When the mapping ψ is an isometry, we expect $\Phi(X)$ to be small. We now take a close look at $\Phi(X)$.

$$\Phi(X) = \sum_{i=1}^{N} \|W_i X_i\|_F^2 = \sum_{i=1}^{N} \left\|W_i X_i^{(1)}\right\|^2 + \sum_{i=1}^{N} \left\|W_i X_i^{(2)}\right\|^2,$$

where $X_i^{(j)}$ is the j-th column of the neighborhood X_i. Define $e_i^{(j)} = \left\|W_i X_i^{(j)}\right\|^2$, and note that $e_i^{(j)}$ depends on the different algorithms through the definition of the matrices W_i. The quantity $e_i^{(j)}$ is the portion of error obtained by using the j-th column of the i-th neighborhood when using the original sample as output. Denote $\bar{e}^{(j)} = \frac{1}{N}\sum_i e_i^{(j)}$, the average error originating from the j-th column.

We define two different embeddings for $\psi(X)$, following the logic of Sec. 4.1. Let

$$Y = X\Sigma^{-1/2} \tag{9}$$

be the first embedding. Note that Y is just the original sample up to a linear transformation that ensures that the normalization constraints $\mathrm{Cov}(Y) = I$ and $Y'\mathbf{1} = \mathbf{0}$ hold. Moreover, Y is the only transformation of X that satisfies these conditions, which are the normalization constraints for LLE, HLLE, and LTSA. In Section 5.2 we discuss the modified embeddings for LEM and DFM.

The second embedding, Z, is given by

$$z_i = \begin{cases} \left(\frac{x_i^{(1)}}{\sigma}, \frac{-x_i^{(1)}}{\rho} - \bar{z}^{(2)}\right) & x_i^{(1)} < 0 \\ \left(\frac{x_i^{(1)}}{\sigma}, \frac{\kappa x_i^{(1)}}{\rho} - \bar{z}^{(2)}\right) & x_i^{(1)} \geq 0 \end{cases}. \tag{10}$$

Here

$$\kappa = \Big(\sum_{i:x_i^{(1)}<0} \left(x_i^{(1)}\right)^2\Big)^{1/2} \Big(\sum_{i:x_i^{(1)}\geq 0} \left(x_i^{(1)}\right)^2\Big)^{-1/2} \tag{11}$$

ensures that $\mathrm{Cov}(Z^{(1)}, Z^{(2)}) = 0$, and $\bar{z}^{(2)} = \frac{1}{N}(\sum_{x_i^{(1)}\geq 0} \frac{\kappa x_i^{(1)}}{\rho} + \sum_{x_i^{(1)}<0} \frac{-x_i^{(1)}}{\rho})$ and ρ are chosen so that the sample mean and variance of $Z^{(2)}$ are equal to zero and one, respectively. We assume without loss of generality that $\kappa \geq 1$.

Note that Z depends only on the first column of X. Moreover, each point z_i is just a linear transformation of $x_i^{(1)}$. In the case of neighborhoods Z_i, the situation can be different. If the first column of X_i is either non-negative or non-positive, then Z_i is indeed a linear transformation of $X_i^{(1)}$. However, if $X_i^{(1)}$ is located on both sides of zero, Z_i is not a linear transformation of $X_i^{(1)}$. Denote by N_0 the set of indices i of neighborhoods Z_i that are not linear transformations of $X_i^{(1)}$. The number $|N_0|$ depends on the number of nearest neighbors K. Recall that for each neighborhood, we defined the radius $r(i) = \max_{j,k\in\{0,\ldots,K\}} \left\|x_{i,j} - x_{i,k}\right\|$. Define $r_{\max} = \max_{i\in N_0} r(i)$ to be the maximum radius of neighborhoods i, such that $i \in N_0$.

5.2 The Embeddings for LEM and DFM

So far we have claimed that given the original sample X, we expect the output to resemble Y (see Eq. 9). However, Y does not satisfy the normalization constraints of Eq. 3 for the cases of LEM and DFM. Define $\hat{Y}$ to be the only affine transformation of X (up to rotation) that satisfies the normalization constraint of LEM and DFM. When the original sample is given by X, we expect the output of LEM and DFM to resemble $\hat{Y}$. We note that unlike the matrix Y that was defined in terms of the matrix X only, $\hat{Y}$ depends also on the choice of neighborhoods through the matrix D that appears in the normalization constraints.

We define $\hat{Y}$ more formally. Denote $\widetilde{X} = X - \frac{1}{\mathbf{1}'D\mathbf{1}}\mathbf{1}\mathbf{1}'DX$. Note that $\widetilde{X}$ is just a translation of X that ensures that $\widetilde{X}'D\mathbf{1} = \mathbf{0}$. The matrix $\widetilde{X}'D\widetilde{X}$ is positive definite and therefore can be presented by $\Gamma\widehat{\Sigma}\Gamma'$ where Γ is a 2×2 orthogonal matrix and

$$\widehat{\Sigma} = \begin{pmatrix} \hat{\sigma}^2 & 0 \\ 0 & \hat{\tau}^2 \end{pmatrix},$$

where $\hat{\sigma} \geq \hat{\tau}$. Define $\widehat{X} = \widetilde{X}\Gamma$; then $\widehat{Y} = \widehat{X}\widehat{\Sigma}^{-1/2}$ is the only affine transformation of X that satisfies the normalization constraints of LEM and DFM; namely, we have $\widehat{Y}'D\widehat{Y} = I$ and $\widehat{Y}'D\mathbf{1} = \mathbf{0}$.

We define $\widehat{Z}$ similarly to Eq. 10,

$$\hat{z}_i = \begin{cases} \left(\frac{\hat{x}_i^{(1)}}{\hat{\sigma}}, \frac{-\hat{x}_i^{(1)}}{\hat{\rho}} - \tilde{z}^{(2)}\right) & \hat{x}_i^{(1)} < 0 \\ \\ \left(\frac{\hat{x}_i^{(1)}}{\hat{\sigma}}, \frac{\hat{\kappa}\hat{x}_i^{(1)}}{\hat{\rho}} - \tilde{z}^{(2)}\right) & \hat{x}_i^{(1)} \geq 0 \end{cases},$$

where $\hat{\kappa}$ is defined by Eq. 11 with respect to $\widehat{X}$, $\tilde{z}^{(2)} = \frac{1}{N}\left(\sum_{x_i^{(1)} \geq 0} \frac{d_{ii}\hat{\kappa}x_i^{(1)}}{\rho} + \sum_{x_i^{(1)} < 0} \frac{-d_{ii}x_i^{(1)}}{\rho}\right)$ and $\hat{\rho}^2 = \kappa^2 \sum_{\hat{x}_i^{(1)} \geq 0} d_{ii}\left(\hat{x}_i^{(1)}\right)^2 + \sum_{\hat{x}_i^{(1)} \leq 0} d_{ii}\left(\hat{x}_i^{(1)}\right)^2$.

A similar analysis to that of Y and Z can be performed for $\widehat{Y}$ and $\widehat{Z}$. The same necessary conditions for success are obtained, with σ, τ, and ρ replaced by $\hat{\sigma}$, $\hat{\tau}$, and $\hat{\rho}$, respectively. In the case where the distribution of the original points is uniform, the ratio $\frac{\hat{\sigma}}{\hat{\tau}}$ is close to the ratio $\frac{\sigma}{\tau}$ and thus the necessary conditions for the success of LEM and DFM are similar to the conditions in Corollary 5.2.

5.3 Characterization of the Embeddings

The main result of this section provides necessary conditions for the success of the normalized-output algorithms. Following Section 3, we say that the algorithms fail if $\Phi(Y) > \Phi(Z)$, where Y and Z are defined in Eqs. 9 and 10, respectively. Thus, a necessary condition for the success of the normalized-output algorithms is that $\Phi(Y) \leq \Phi(Z)$.

Theorem 5.1 *Let X be a sample from a two-dimensional domain and let $\psi(X)$ be its embedding in high-dimensional space. Let Y and Z be defined as above. Then*

$$\frac{\kappa^2}{\rho^2}\left(\bar{e}^{(1)} + \frac{|N_0|}{N} c_a r_{\max}^2\right) < \frac{\bar{e}^{(2)}}{\tau^2} \quad \Longrightarrow \quad \Phi(Y) > \Phi(Z)\,, \tag{12}$$

where c_a is a constant that depends on the specific algorithm. For the algorithms LEM and DFM a more restrictive condition can be defined:

$$\frac{\kappa^2}{\rho^2}\bar{e}^{(1)} < \frac{\bar{e}^{(2)}}{\tau^2} \quad \Longrightarrow \quad \Phi(Y) > \Phi(Z)\,.$$

For the proof, see Appendix A.6.

Note that the bound in Eq. 12 depends on the radii of the neighborhoods, and when the maximum radius is large, the bound is less effective. However, there is a tradeoff between enlarging the radius and improving the description of the neighborhoods, that is, reducing $\bar{e}^{(2)}$. In other words, when the neighborhoods are large, one can expect a large average error in the description of the neighborhoods, since the Euclidian approximation of the neighborhoods is less accurate for neighborhoods of large radius.

Adding some assumptions, we can obtain a simpler criterion. First note that, in general, $\bar{e}^{(1)}$ and $\bar{e}^{(2)}$ should be of the same order, since it can be assumed that, locally, the neighborhoods are

uniformly distributed. Second, following Lemma A.2 (see Appendix A.8), when $X^{(1)}$ is a sample from a symmetric unimodal distribution it can be assumed that $\kappa \approx 1$ and $\rho^2 > \frac{\sigma^2}{8}$. Then we have the following corollary:

Corollary 5.2 *Let X,Y,Z be as in Theorem 5.1. Let $c = \sigma/\tau$ be the ratio between the variance of the first and second columns of X. Assume that $\bar{e}^{(1)} < \sqrt{2}\bar{e}^{(2)}$, $\kappa < \sqrt[4]{2}$, and $\rho^2 > \frac{\sigma^2}{8}$. Then*

$$4\left(1+\frac{|N_0|}{N}\frac{c_a r_{\max}^2}{\sqrt{2}\bar{e}^{(2)}}\right) < c \Rightarrow \Phi(Y) > \Phi(Z)\,.$$

For LEM and DFM, we can write

$$4 < c \Rightarrow \Phi(Y) > \Phi(Z)\,.$$

We emphasize that both Theorem 5.1 and Corollary 5.2 do not state that Z is the output of the normalized-output algorithms. However, when the difference between the right side and the left side of the inequalities is large, one cannot expect the output to resemble the original sample (see Lemma 3.1). In these cases we say that the algorithms fail to recover the structure of the original domain.

5.4 Generalization of the Results to Manifolds of Higher Dimensions

The discussion above introduced necessary conditions for the normalized-output algorithms' success on two-dimensional manifolds embedded in $\mathbb{R}^{\mathcal{D}}$. Necessary conditions for success on general d-dimensional manifolds, $d \geq 3$, can also be obtained. We present here a simple criterion to demonstrate the fact that there are d-dimensional manifolds that the normalized-output algorithms cannot recover.

Let $X = [X^{(1)},\ldots,X^{(d)}]$ be a $N \times d$ sample from a d-dimensional domain. Assume that the input for the normalized-output algorithms is given by $\psi(X) \subset \mathbb{R}^{\mathcal{D}}$ where $\psi : \mathbb{R}^d \to \mathbb{R}^{\mathcal{D}}$ is a smooth function and $\mathcal{D} \geq d$ is the dimension of the input. We assume without loss of generality that $X'\mathbf{1} = \mathbf{0}$ and that $\mathrm{Cov}(X)$ is a diagonal matrix. Let $Y = X\mathrm{Cov}(X)^{-1/2}$. We define the matrix $Z = [Z^{(1)},\ldots,Z^{(d)}]$ as follows. The first column of Z, $Z^{(1)}$, equals the first column of Y, namely, $Z^{(1)} = Y^{(1)}$. We define the second column $Z^{(2)}$ similarly to the definition in Eq. 10:

$$Z_i^{(2)} = \begin{cases} \frac{-x_i^{(1)}}{\rho} - \bar{z}^{(2)} & x_i^{(1)} < 0 \\ \\ \frac{\kappa x_i^{(1)}}{\rho} - \bar{z}^{(2)} & x_i^{(1)} \geq 0 \end{cases}, \tag{13}$$

where κ is defined as in Eq. 11, and $\bar{z}^{(2)}$ and ρ are chosen so that the sample mean and variance of $Z^{(2)}$ are equal to zero and one, respectively. We define the next $d-2$ columns of Z by

$$Z^{(j)} = \frac{Y^{(j)} - \sigma_{2j}Z^{(2)}}{\sqrt{1-\sigma_{2j}^2}}; \quad j = 3,\ldots,d\,,$$

where $\sigma_{2j} = Z^{(2)\prime}Y^{(j)}$. Note that $Z'\mathbf{1} = \mathbf{0}$ and $\mathrm{Cov}(Z) = I$. Denote $\sigma_{\max} = \max_{j\in\{3,\ldots,d\}}\sigma_{2j}$.

We bound $\Phi(Z)$ from above:

$$\begin{aligned}\Phi(Z) &= \Phi(Y^{(1)})+\Phi(Z^{(2)})+\sum_{i=1}^{N}\left(\frac{1}{1-\sigma_{2j}^2}\right)\sum_{j=3}^{d}\left\|W_i\left(Y_i^{(j)}-\sigma_{2j}Z_i^{(2)}\right)\right\|^2 \\ &\leq \Phi(Y^{(1)})+\Phi(Z^{(2)})+\frac{1}{1-\sigma_{\max}^2}\sum_{i=1}^{N}\sum_{j=3}^{d}\left\|W_iY_i^{(j)}\right\|^2+\frac{\sigma_{\max}^2}{1-\sigma_{\max}^2}\sum_{i=1}^{N}\sum_{j=3}^{d}\left\|W_iZ_i^{(2)}\right\|^2 \\ &= \Phi(Y^{(1)})+\frac{1+(d-3)\sigma_{\max}^2}{1-\sigma_{\max}^2}\Phi(Z^{(2)})+\frac{1}{1-\sigma_{\max}^2}\sum_{j=3}^{d}\Phi(Y^{(j)}).\end{aligned}$$

Since we may write $\Phi(Y)=\sum_{j=1}^{d}\Phi(Y^{(j)})$, we have

$$\frac{1+(d-3)\sigma_{\max}^2}{1-\sigma_{\max}^2}\Phi(Z^{(2)})<\Phi(Y^{(2)})+\frac{\sigma_{\max}^2}{1-\sigma_{\max}^2}\sum_{j=3}^{d}\Phi(Y^{(j)})\Rightarrow\Phi(Z)<\Phi(Y).$$

When the sample is taken from a symmetric distribution with respect to the axes, one can expect $\sigma_{\max}$ to be small. To see this, note that by symmetry and Eq. 13, $Z_i^{(2)}\approx|Y_i^{(1)}|$, and by assumption $\mathrm{Cov}(Y^{(1)},Y^{(j)})=0$ for $j=3,\ldots,d$. Hence, by the symmetry of $Y^{(j)}$, σ_{2j} is expected to be small. In the specific case of the d-dimensional grid, $\sigma_{\max}=0$. Indeed, $Y^{(j)}$ is symmetric around zero, and all values of $Z^{(2)}$ appear for a given value of $Y^{(j)}$. Hence, both LEM and DFM are expected to fail whenever the ratio between the length of the grid in the first and second coordinates is slightly greater than 2 or more, regardless of the length of grid in the other coordinates, similar to the result presented in Theorem 4.1. Corresponding results for the other normalized-output algorithms can also be obtained, similar to the derivation of Corollary 5.2.

5.5 Numerical Results

We ran all five normalized-output algorithms, along with Isomap, on three data sets. We used the Matlab implementations written by the algorithms' authors as provided by Wittman (retrieved Jan. 2007).

The first data set is a 1600-point sample from the swissroll as obtained from Wittman (retrieved Jan. 2007). The results for the swissroll are given in Fig. 7, A1-F1. The results for the same swissroll, after its first dimension was stretched by a factor 3, are given in Fig. 7, A2-F2. The original and stretched swissrolls are presented in Fig. 6A. The results for $K=8$ are given in Fig. 7. We also checked for $K=12,16$; but "short-circuits" occur (see Balasubramanian et al., 2002, for a definition and discussion of "short-circuits").

The second data set consists of 2400 points, uniformly sampled from a "fishbowl", where a "fishbowl" is a two-dimensional sphere minus a neighborhood of the northern pole (see Fig. 6B for both the "fishbowl" and its stretched version). The results for $K=8$ are given in Fig. 8. We also checked for $K=12,16$; the results are roughly similar. Note that the "fishbowl" is a two-dimensional manifold embedded in $\mathbb{R}^3$, which is not an isometry.

The third data set consists of 900 images of the globe, each of 100×100 pixels (see Fig. 6C). The images, provided by Hamm et al. (2005), were obtained by changing the globe's azimuthal and elevation angles. The parameters of the variations are given by a 30×30 array that contains -45 to 45 degrees of azimuth and -30 to 60 degrees of elevation. We checked the algorithms both on the

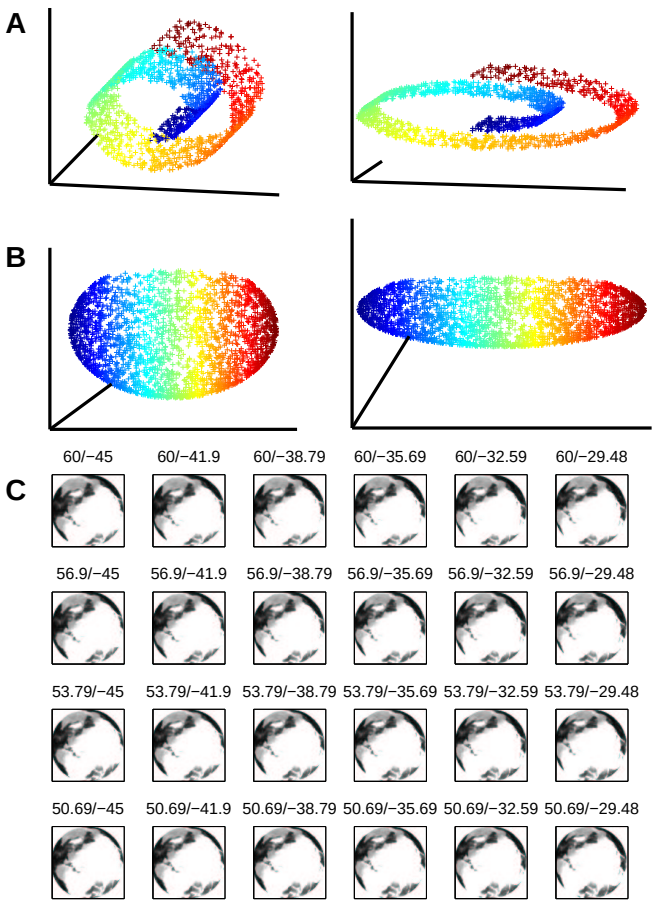


Figure 6: The data sets for the first example appear in panel A. In the left appears the 1600-point original swissroll and in the right appears the same swissroll, after its first dimension was stretched by a factor of 3. The data for the second example appear in panel B. In the left appears a 2400-point uniform sample from the "fishbowl", and in the right appears the same "fishbowl", after its first dimension was stretched by a factor of 4. In panel C appears the upper left corner of the array of 100×100 pixel images of the globe. Above each image we write the elevation and azimuth.

entire set of images and on a strip of 30×10 angular variations. The results for $K = 8$ are given in Fig. 9. We also checked for $K = 12, 16$; the results are roughly similar.

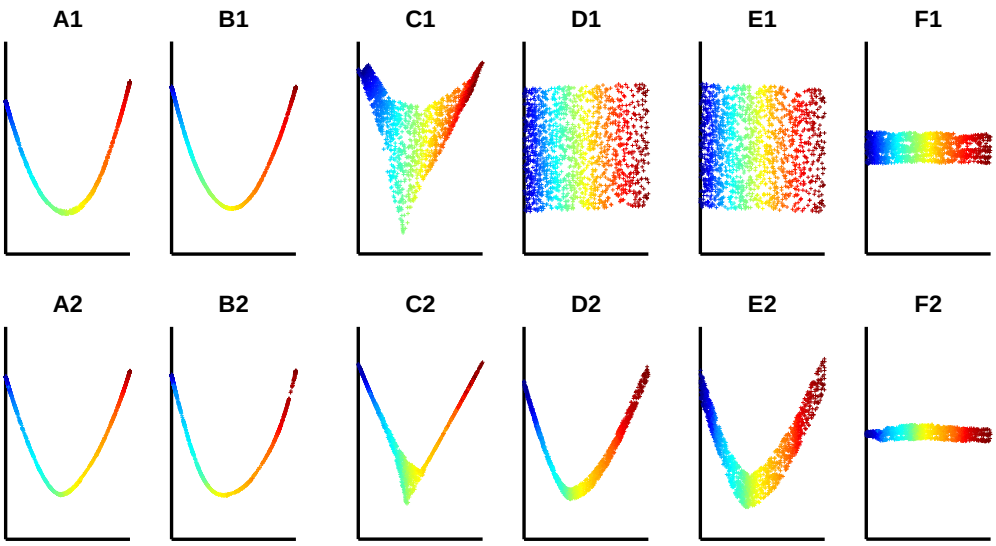


Figure 7: The output of LEM on 1600 points sampled from a swissroll is presented in A1. The output of LEM on the same swissroll after stretching its first dimension by a factor of 3 is presented in A2. Similarly, the outputs of DFM, LLE, LTSA, HLLE, and Isomap are presented in B1-2, C1-2, D1-2, E1-2, and F1-2, respectively. We used $K = 8$ for all algorithms except DFM, where we used $\sigma = 2$.

These three examples, in addition to the noisy version of the two-dimensional strip discussed in Section 3 (see Fig. 1), clearly demonstrate that when the aspect ratio is sufficiently large, all the normalized-output algorithms prefer to collapse their output.

6. Asymptotics

In the previous sections we analyzed the phenomenon of global distortion obtained by the normalized-output algorithms on a finite input sample. However, it is of great importance to explore the limit behavior of the algorithms, that is, the behavior when the number of input points tends to infinity. We consider the question of convergence in the case of input that consists of a d-dimensional manifold embedded in $\mathbb{R}^{\mathcal{D}}$, where the manifold is isometric to a convex subset of Euclidean space. By convergence we mean recovering the original subset of $\mathbb{R}^d$ up to a non-singular affine transformation.

Some previous theoretical works presented results related to the convergence issue. Huo and Smith (2006) proved convergence of LTSA under some conditions. However, to the best of our knowledge, no proof or contradiction of convergence has been given for any other of the normalized-output algorithms. In this section we discuss the various algorithms separately. We also discuss the influence of noise on the convergence. Using the results from previous sections, we show that there

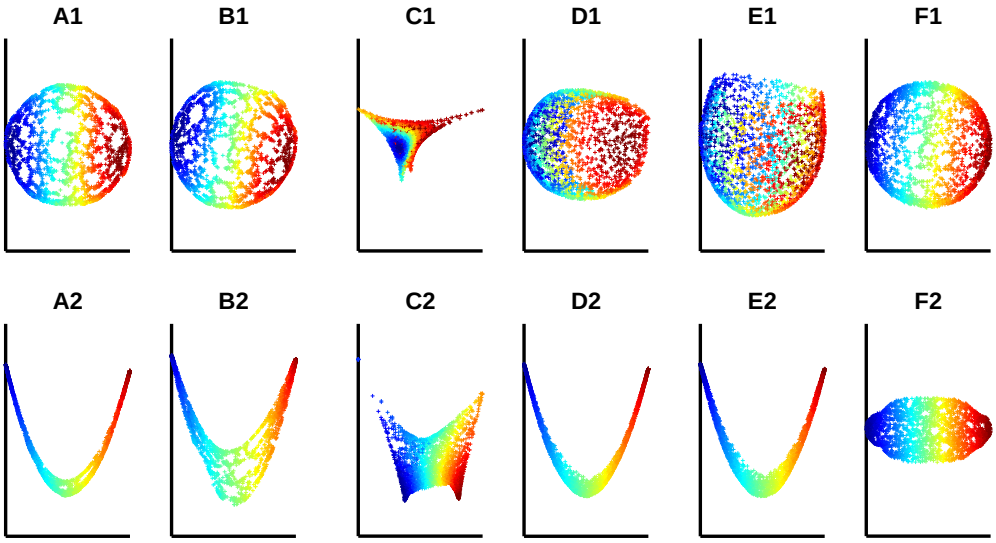


Figure 8: The output of LEM on 2400 points sampled from a "fishbowl" is presented in A1. The output of LEM on the same "fishbowl" after stretching its first dimension by a factor of 4 is presented in A2. Similarly, the outputs of DFM, LLE, LTSA, HLLE, and Isomap are presented in B1-2, C1-2, D1-2, E1-2, and F1-2, respectively. We used $K = 8$ for all algorithms except DFM, where we used $\sigma = 2$.

are classes of manifolds on which the normalized-output algorithms cannot be expected to recover the original sample, not even asymptotically.

6.1 LEM and DFM

Let $x_1, x_2, \ldots$ be a uniform sample from the two-dimensional strip $S = [0, L] \times [0, 1]$. Let $X_n = [x_1, \ldots, x_n]'$ be the sample of size n. Let $K = K(n)$ be the number of nearest neighbors. Then when $K \ll n$ there exists with probability one an N, such that for all $n > N$ the assumptions of Corollary 5.2 hold. Thus, if $L > 4$ we do not expect either LEM or DFM to recover the structure of the strip as the number of points in the sample tends to infinity. Note that this result does not depend on the number of neighbors or the width of the kernel, which can be changed as a function of the number of points n, as long as $K \ll n$. Hence, we conclude that LEM and DFM generally do not converge, regardless of the choice of parameters.

In the rest of this subsection we present further explanations regarding the failure of LEM and DFM based on the asymptotic behavior of the graph Laplacian (see Belkin and Niyogi, 2003, for details). Although it was not mentioned explicitly in this paper, it is well known that the outputs of LEM and DFM are highly related to the lower non-negative eigenvectors of the normalized graph Laplacian matrix (see Appendix A.1). It was shown by Belkin and Niyogi (2005), Hein et al. (2005), and Singer (2006) that the graph Laplacian operator converges to the continuous Laplacian

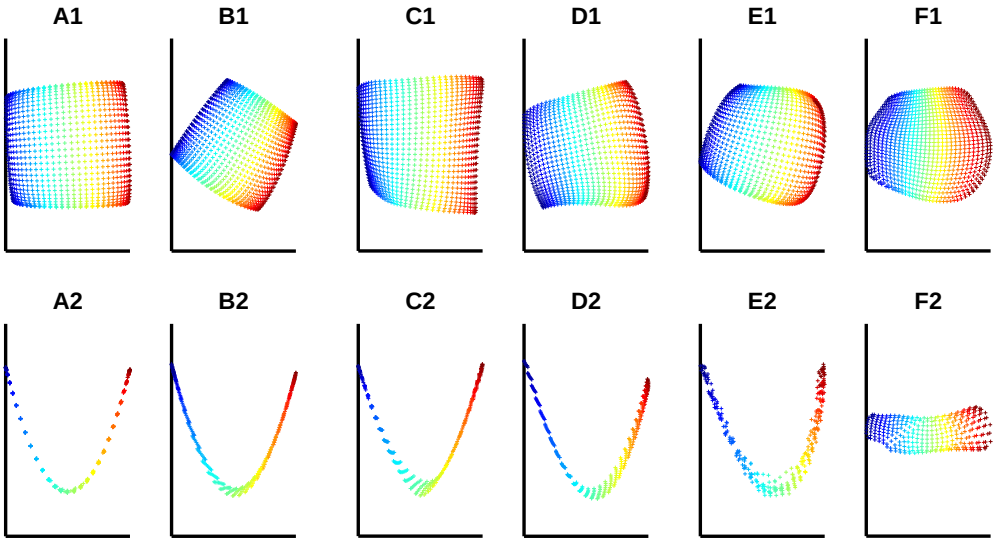


Figure 9: The output of LEM on the 30×30 array of the globe rotation images is presented in A1; the output of LEM on the array of 30×10 is presented in A2. Similarly, the outputs of DFM, LLE, LTSA, HLLE, and Isomap are presented in B1-2, C1-2, D1-2, E1-2, and F1-2 respectively. We used $K = 8$ for all algorithms except DFM, where we chose σ to be the root of the average distance between neighboring points.

operator. Thus, taking a close look at the eigenfunctions of the continuous Laplacian operator may reveal some additional insight into the behavior of both LEM and DFM.

Our working example is the two-dimensional strip $S = [0,L] \times [0,1]$, which can be considered as the continuous counterpart of the grid X defined in Section 4. Following Coifman and Lafon (2006) we impose the Neumann boundary condition (see details therein). The eigenfunctions $\varphi_{i,j}(x_1,x_2)$ and eigenvalues $\lambda_{i,j}$ on the strip S under these conditions are given by

$$\varphi_{i,j}(x_1,x_2) = \cos\left(\frac{i\pi}{L}x_1\right)\cos(j\pi x_2) \quad \lambda_{i,j} = \left(\frac{i\pi}{L}\right)^2 + (j\pi)^2 \quad \text{for } i,j = 0,1,2,\ldots.$$

When the aspect ratio of the strip satisfies $L > M \in \mathbb{N}$, the first M non-trivial eigenfunctions are $\varphi_{i,0}$, $i = 1,\ldots,M$, which are functions only of the first variable x_1. Any embedding of the strip based on the first M eigenfunctions is therefore a function of only the first variable x_1. Specifically, whenever $L > 2$ the two-dimensional embedding is a function of the first variable only, and therefore clearly cannot establish a faithful embedding of the strip. Note that here we have obtained the same ratio constant $L > 2$ computed for the grid (see Section 4 and Figs. 4 and 5) and not the looser constant $L > 4$ that was obtained in Corollary 5.2 for general manifolds.

6.2 LLE, LTSA and HLLE

As mentioned in the beginning of this section, Huo and Smith (2006) proved the convergence of the LTSA algorithm. The authors of HLLE proved that the continuous manifold can be recovered by finding the null space of the continuous Hessian operator (see Donoho and Grimes, 2004, Corollary). However, this is not a proof that the algorithm HLLE converges. In the sequel, we try to understand the relation between Corollary 5.2 and the convergence proof of LTSA.

Let $x_1, x_2, \ldots$ be a sample from a compact and convex domain Ω in $\mathbb{R}^2$. Let $X_n = [x_1, \ldots, x_n]'$ be the sample of size n. Let ψ be an isometric mapping from $\mathbb{R}^2$ to $\mathbb{R}^{\mathcal{D}}$, where $\mathcal{D} > 2$. Let $\psi(X_n)$ be the input for the algorithms. We assume that there is an N such that for all $n > N$ the assumptions of Corollary 5.2 hold. This assumption is reasonable, for example, in the case of a uniform sample from the strip S. In this case Corollary 5.2 states that $\Phi(Z_n) < \Phi(Y_n)$ whenever

$$4\left(1 + \frac{|n_0|}{n}\frac{c_a r_{\max,n}^2}{\sqrt{2}\bar{e}_n^{(2)}}\right) < c_n\,,$$

where c_n is the ratio between the variance of $X_n^{(1)}$ and $X_n^{(2)}$ assumed to converge to a constant c. The expression $\frac{|n_0|}{n}$ is the fraction of neighborhoods $X_{i,n}$ such that $X_{i,n}^{(1)}$ is located on both sides of zero. $r_{\max,n}$ is the maximum radius of neighborhood in n_0. Note that we expect both $\frac{|n_0|}{n}$ and $r_{\max,n}$ to be bounded whenever the radius of the neighborhoods does not increase. Thus, Corollary 5.2 tells us that if $\{\bar{e}_n^{(2)}\}$ is bounded from below, we cannot expect convergence from LLE, LTSA or HLLE when c is large enough.

The consequence of this discussion is that a necessary condition for the convergence of LLE, LTSA and HLLE is that $\{\bar{e}_n^{(2)}\}$ (and hence, from the assumptions of Corollary 5.2, also $\{\bar{e}_n^{(1)}\}$) converges to zero. If the two-dimensional manifold $\psi(\Omega)$ is not contained in a linear two-dimensional subspace of $\mathbb{R}^{\mathcal{D}}$, the mean error $\bar{e}_n^{(2)}$ is typically not zero due to curvature. However, if the radii of the neighborhoods tend to zero while the number of points in each neighborhood tends to infinity, we expect $\bar{e}_n^{(2)} \to 0$ for both LTSA and HLLE. This is because the neighborhood matrices W_i are based on the linear approximation of the neighborhood as captured by the neighborhood SVD. When the radius of the neighborhood tends to zero, this approximation gets better and hence the error tends to zero. The same reasoning works for LLE, although the use of regularization in the second step of LLE may prevent $\bar{e}_n^{(2)}$ from converging to zero (see Section 2).

We conclude that a necessary condition for convergence is that the radii of the neighborhoods tend to zero. In the presence of noise, this requirement cannot be fulfilled. Assume that each input point is of the form $\psi(x_i) + \varepsilon_i$ where $\varepsilon_i \in \mathbb{R}^{\mathcal{D}}$ is a random error that is independent of ε_j for $j \neq i$. We may assume that $\varepsilon_i \sim N(0, \alpha^2 I)$, where α is a small constant. If the radius of the neighborhood is smaller than α, the neighborhood cannot be approximated reasonably by a two-dimensional projection. Hence, in the presence of noise of a constant magnitude, the radii of the neighborhoods cannot tend to zero. In that case, LLE, LTSA and HLLE might not converge, depending on the ratio c. This observation seems to be known also to Huo and Smith (2006), who wrote:

> "... we assume $\alpha = o(r)$; that is, we have $\frac{\alpha}{r} \to 0$, as $r \to 0$.
>
> It is reasonable to require that the error bound (α) be smaller than the size of the neighborhood (r), which is reflected in the above condition. Notice that this condition is also

somewhat nonstandard, since the magnitude of the errors is assumed to depend on n, but it seems to be necessary to ensure the consistency of LTSA."[2]

Summarizing, convergence may be expected when $n \to \infty$, if no noise is introduced. If noise is introduced and if σ/τ is large enough (depending on the level of noise α), convergence cannot be expected (see Fig. 1).

7. Concluding Remarks

In the introduction to this paper we posed the following question: Do the normalized-output algorithms succeed in revealing the underlying low-dimensional structure of manifolds embedded in high-dimensional spaces? More specifically, does the output of the normalized-output algorithms resemble the original sample up to affine transformation?

The answer, in general, is no. As we have seen, Theorem 5.1 and Corollary 5.2 show that there are simple low-dimensional manifolds, isometrically embedded in high-dimensional spaces, for which the normalized-output algorithms fail to find the appropriate output. Moreover, the discussion in Section 6 shows that when noise is introduced, even of small magnitude, this result holds asymptotically for all the normalized-output algorithms. We have demonstrated these results numerically for four different examples: the swissroll, the noisy strip, the (non-isometrically embedded) "fishbowl", and a real-world data set of globe images. Thus, we conclude that the use of the normalized-output algorithms on arbitrary data can be problematic.

The main challenge raised by this paper is the need to develop manifold-learning algorithms that have low computational demands, are robust against noise, and have theoretical convergence guarantees. Existing algorithms are only partially successful: normalized-output algorithms are efficient, but are not guaranteed to converge, while Isomap is guaranteed to converge, but is computationally expensive. A possible way to achieve all of the goals simultaneously is to improve the existing normalized-output algorithms. While it is clear that, due to the normalization constraints, one cannot hope for geodesic distances preservation nor for neighborhoods structure preservation, success as measured by other criteria may be achieved. A suggestion of improvement for LEM appears in Gerber et al. (2007), yet this improvement is both computationally expensive and lacks a rigorous consistency proof. We hope that future research finds additional ways to improve the existing methods, given the improved understanding of the underlying problems detailed in this paper.

Acknowledgments

We are grateful to the anonymous reviewers of present and earlier versions of this manuscript for their helpful suggestions. We thank an anonymous referee for pointing out errors in the proof of Lemma 3.1. We thank J. Hamm for providing the database of globe images. This research was supported in part by Israeli Science Foundation grant and in part by NSF, grant DMS-0605236.

2. We replaced the original τ and σ with r and α respectively to avoid confusion with previous notations.

Appendix A. Detailed Proofs and Discussions

This section contains detailed proofs of Equations 4 and 7, Lemmas 3.1 and A.2, and Theorem 5.1. It also contains discussions regarding the equivalence of the normalized-output algorithms' representations, and the estimation of $F(K)$ and $\widetilde{F}(K)$ for a ball of radius r (see Section 4).

A.1 The Equivalence of the Algorithms' Representations

For LEM, note that according to our representation, one needs to minimize

$$\Phi(Y) = \sum_{i=1}^{N} \|W_i Y_i\|_F^2 = \sum_{i=1}^{N} \sum_{j=1}^{K} w_{i,j} \|y_i - y_{i,j}\|^2,$$

under the constraints $Y'D\mathbf{1} = \mathbf{0}$ and $Y'DY = I$. Define $\hat{w}_{rs} = w_{r,j}$ if s is the j-th neighbor of r and zero otherwise. Define $\hat{D}$ to be the diagonal matrix such that $d_{rr} = \sum_{s=1}^{N} \hat{w}_{rs}$; note that $\hat{D} = D$. Using these definitions, one needs to minimize $\Phi(Y) = \sum_{r,s} \hat{w}_{rs} \|y_r - y_s\|^2$ under the constraints $Y'\hat{D}\mathbf{1} = \mathbf{0}$ and $Y'\hat{D}Y = I$, which is the the authors' representation of the algorithm.

For DFM, as for LEM, we define the weights $\hat{w}_{rs}$. Define the $N \times N$ matrix $\hat{W} = (\hat{w}_{rs})$. Define the matrix $D^{-1}\hat{W}$; note that this matrix is a Markovian matrix and that $v^{(0)} \equiv \mathbf{1}$ is its eigenvector corresponding to eigenvalue 1, which is the largest eigenvalue of the matrix. Let $v^{(p)}$, $p = 1, \ldots, d$ be the next d eigenvectors, corresponding to the next d largest eigenvalues λ_p, normalized such that $v^{(p)\prime} D v^{(p)} = 1$. Note that the vectors $v^{(0)}, \ldots, v^{(d)}$ are also the eigenvectors of $I - D^{-1}W$ corresponding to the $d+1$ lowest eigenvalues. Thus, the matrix $[v^{(1)}, \ldots, v^{(d)}]$ (up to rotation) can be computed by minimizing $\operatorname{tr}(Y'(D-W)Y)$ under the constraints $Y'DY = I$ and $Y'D\mathbf{1} = \mathbf{0}$. Simple computation shows (see Belkin and Niyogi, 2003, Eq. 3.1) that $\operatorname{tr}(Y'(D-W)Y) = \frac{1}{2}\sum_{r,s} \hat{w}_{rs} \|y_r - y_s\|^2$. We already showed that $\Phi(Y) = \sum_{r,s} \hat{w}_{rs} \|y_r - y_s\|^2$. Hence, minimizing $\operatorname{tr}(Y'(D-W)Y)$ under the constraints $Y'DY = I$ and $Y'D\mathbf{1} = \mathbf{0}$ is equivalent to minimizing $\Phi(Y)$ under the same constraints. The embedding suggested by Coifman and Lafon (2006) (up to rotation) is the matrix $\left[\lambda_1 \frac{v^{(1)}}{\|v^{(1)}\|}, \ldots, \lambda_d \frac{v^{(d)}}{\|v^{(d)}\|}\right]$. Note that this embedding can be obtained from the output matrix Y by a simple linear transformation.

For LLE, note that according to our representation, one needs to minimize

$$\Phi(Y) = \sum_{i=1}^{N} \|W_i Y_i\|_F^2 = \sum_{i=1}^{N} \|y_i - \sum_{j=1}^{K} w_{i,j} y_{i,j}\|^2$$

under the constraints $Y'\mathbf{1} = \mathbf{0}$ and $\operatorname{Cov}(Y) = I$, which is the minimization problem given by Roweis and Saul (2000).

The representation of LTSA is similar to the representation that appears in the original paper, differing only in the weights' definition. We defined the weights W_i following Huo and Smith (2006), who showed that both definitions are equivalent.

For HLLE, note that according to our representation, one needs to minimize

$$\Phi(Y) = \sum_{i=1}^{N} \|W_i Y_i\|_F^2 = \sum_{i=1}^{N} \operatorname{tr}\left(Y_i' H_i' H_i Y_i\right)$$

under the constraint $\operatorname{Cov}(Y) = I$. This is equivalent (up to a multiplication by $\sqrt{(N)}$) to minimizing $\operatorname{tr}(Y'\mathcal{H}Y)$ under the constraint $Y'Y = I$, where $\mathcal{H}$ is the matrix that appears in the original definition

of the algorithm. This minimization can be calculated by finding the $d+1$ lowest eigenvectors of $\mathcal{H}$, which is the embedding suggested by Donoho and Grimes (2004).

A.2 Proof of Lemma 3.1

We begin by estimating $\Phi(\widetilde{Y})$.

$$\begin{aligned}
\Phi(\widetilde{Y}) &= \sum_{i=1}^{N} \|W_i Y_i + \varepsilon W_i E_i\|_F^2 = \sum_{i=1}^{N}\sum_{j=0}^{K} \left\|W_i y_{i,j} + \varepsilon W_i e_{i,j}\right\|^2 \qquad (14)\\
&\geq \sum_{i=1}^{N}\sum_{j=0}^{K} \left(\left\|W_i y_{i,j}\right\|^2 - 2\varepsilon |(W_i y_{i,j})' W_i e_{i,j}|\right)\\
&\geq \sum_{i=1}^{N}\sum_{j=0}^{K} \left((1-\varepsilon)\left\|W_i y_{i,j}\right\|^2 - \varepsilon \left\|W_i e_{i,j}\right\|^2\right)\\
&= (1-\varepsilon)\sum_{i=1}^{N} \|W_i Y_i\|_F^2 - \varepsilon \sum_{i=1}^{N} \|W_i E_i\|_F^2\\
&\geq (1-\varepsilon)\Phi(Y) - \varepsilon \sum_{i=1}^{N} \|W_i\|_F^2 \|E_i\|_F^2 ,
\end{aligned}$$

where $e_{i,j}$ denotes the j-th row of E_i.

We bound $\|W_i\|_F^2$ for each of the algorithms by a constant C_a. It can be shown that for LEM and DFM, $C_a \leq 2K$; for LTSA, $C_a \leq K$; for HLLE $C_a \leq \frac{d(d+1)}{2}$. For LLE in the case of positive weights $w_{i,j}$, we have $C_a \leq 2$. Thus, substituting C_a in Eq. 14, we obtain

$$\begin{aligned}
\Phi(\widetilde{Y}) &\geq (1-\varepsilon)\Phi(Y) - \varepsilon C_a \sum_{i=1}^{N}\sum_{j=0}^{K} \left\|e_{i,j}\right\|^2\\
&\geq (1-\varepsilon)\Phi(Y) - \varepsilon C_a S \|E\|_F^2 = (1-\varepsilon)\Phi(Y) - \varepsilon C_a S .
\end{aligned}$$

The last inequality holds true since S is the maximum number of neighborhoods to which a single observation belongs.

A.3 Proof of Equation 4

By definition $\sigma^2 = \mathrm{Var}(X^{(1)})$ and hence,

$$\begin{aligned}
\sigma^2 &= \frac{1}{N}\sum_{i=-m}^{m}\sum_{j=-q}^{q} \left(x_{ij}^{(1)}\right)^2\\
&= \frac{1}{(2m+1)(2q+1)}\sum_{i=-m}^{m}\sum_{j=-q}^{q} i^2\\
&= \frac{2}{2m+1}\sum_{i=1}^{m} i^2\\
&= \frac{2}{2m+1}\frac{(2m+1)(m+1)m}{6}\\
&= \frac{(m+1)m}{3} .
\end{aligned}$$

The computation for τ is similar.

A.4 Estimation of $F(K)$ and $\widetilde{F}(K)$ for a Ball of Radius r

Calculation of $\phi(Y_{ij})$ for general K can be different for different choices of neighborhoods. Therefore, we restrict ourselves to estimating $\phi(Y_{ij})$ when the neighbors are all the points inside an r-ball in the original grid. Recall that $\phi(Y_{ij})$ for an inner point is equal to the sum of the squared distance between y_{ij} and its neighbors. The function

$$f(x_1,x_2) = \left(\frac{x_1}{\sigma}\right)^2 + \left(\frac{x_2}{\tau}\right)^2$$

agrees with the squared distance for points on the grid, where x_1 and x_2 indicate the horizontal and vertical distances from x_{ij} in the original grid, respectively. We estimate $\phi(Y_{ij})$ using integration of $f(x_1,x_2)$ on $B(r)$, a ball of radius r, which yields

$$\phi(Y_{ij}) \approx \int\limits_{(x_1^2+x_2^2)<r^2} f(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = \frac{\pi r^4}{4}\left(\frac{1}{\sigma^2}+\frac{1}{\tau^2}\right). \tag{15}$$

Thus, we obtain $F(K) \approx \frac{\pi r^4}{4}$.

We estimate $\phi(Z_{ij})$ similarly. We define the continuous version of the squared distance in the case of the embedding Z by

$$g(x_1,x_2) = x_1^2\left(\frac{1}{\sigma^2}+\frac{4}{\rho^2}\right).$$

Integration yields

$$\phi(Z_{ij}) \approx \int\limits_{(x_1^2+x_2^2)<r^2} g(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = \frac{\pi r^4}{4}\left(\frac{1}{\sigma^2}+\frac{4}{\rho^2}\right). \tag{16}$$

Hence, we obtain $\widetilde{F}(K) \approx \frac{\pi r^4}{4}$ and the relations between Eqs. 5 and 6 are preserved for a ball of general radius.

For DFM, a general rotation-invariant kernel was considered for the weights. As with Eqs. 15 and 16, the approximations of $\phi(Y_{ij})$ and $\phi(Z_{ij})$ for the general case with neighborhood radius r are given by

$$\int\limits_{(x_1^2+x_2^2)<r^2} f(x_1,x_2)k(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = \left(\pi\int\limits_{0<t<r} k(t^2)t^3\mathrm{d}t\right)\left(\frac{1}{\sigma^2}+\frac{1}{\tau^2}\right)$$

and

$$\int\limits_{(x_1^2+x_2^2)<r^2} g(x_1,x_2)k(x_1,x_2)\mathrm{d}x_1\mathrm{d}x_2 = \left(\pi\int\limits_{0<t<r} k(t^2)t^3\mathrm{d}t\right)\left(\frac{1}{\sigma^2}+\frac{4}{\rho^2}\right).$$

Note that the ratio between these approximations of $\phi(Y_{ij})$ and $\phi(Z_{ij})$ is preserved. In light of these computations it seems that for the general case of rotation-invariant kernels, $\phi(Y_{ij}) > \phi(Z_{ij})$ for aspect ratio sufficiently greater than 2.

A.5 Proof of Equation 7

Direct computation shows that

$$\bar{z}^{(2)} = \frac{(2q+1)2}{N\rho}\sum_{i=1}^{m}(2i) = \frac{2m(m+1)}{(2m+1)\rho}\,.$$

Recall that by definition ρ ensures that $\mathrm{Var}(Z^{(2)}) = 1$. Hence,

$$\begin{aligned} 1 &= \frac{1}{N}\sum_{i=-m}^{m}\sum_{j=-q}^{q}\frac{(2i)^2}{\rho^2} - \left(\bar{z}^{(2)}\right)^2 \\ &= \frac{2}{2m+1}\frac{4m(m+1)(2m+1)}{6\rho^2} - \frac{4m^2(m+1)^2}{(2m+1)^2\rho^2} \\ &= \frac{4m(m+1)}{3\rho^2} - \frac{4m^2(m+1)^2}{(2m+1)^2\rho^2}\,. \end{aligned}$$

Further computation shows that

$$(m+1)m > \frac{4(m+1)^2m^2}{(2m+1)^2}\,.$$

Hence,

$$\rho^2 > \frac{4(m+1)m}{3} - (m+1)m = \sigma^2\,.$$

A.6 Proof of Theorem 5.1

The proof consists of computing $\Phi(Y)$ and bounding $\Phi(Z)$ from above. We start by computing $\Phi(Y)$.

$$\begin{aligned} \Phi(Y) &= \sum_{i=1}^{N}\|W_iY_i\|_F^2 = \sum_{i=1}^{N}\left\|W_iX_i^{(1)}/\sigma\right\|^2 + \sum_{i=1}^{N}\left\|W_iX_i^{(2)}/\tau\right\|^2 \\ &= N\frac{\bar{e}^{(1)}}{\sigma^2} + N\frac{\bar{e}^{(2)}}{\tau^2}\,. \end{aligned}$$

The computation of $\Phi(Z)$ is more delicate because it involves neighborhoods Z_i that are not linear transformations of their original sample counterparts.

$$\begin{aligned} \Phi(Z) &= \sum_{i=1}^{N}\|W_iZ_i\|_F^2 = \sum_{i=1}^{N}\left\|W_iZ_i^{(1)}\right\|^2 + \sum_{i=1}^{N}\left\|W_iZ_i^{(2)}\right\|^2 \\ &= N\frac{\bar{e}^{(1)}}{\sigma^2} + \sum_{i:x_i^{(1)}<0\,,\,i\notin N_0}\left\|W_iX_i^{(1)}/\rho\right\|^2 + \sum_{i:x_i^{(1)}>0\,,\,i\notin N_0}\left\|\kappa W_iX_i^{(1)}/\rho\right\|^2 + \sum_{i\in N_0}\left\|W_iZ_i^{(2)}\right\|^2 \qquad (17) \\ &< N\frac{\bar{e}^{(1)}}{\sigma^2} + N\frac{\kappa^2\bar{e}^{(1)}}{\rho^2} + \sum_{i\in N_0}\left\|W_iZ_i^{(2)}\right\|^2\,. \qquad (18) \end{aligned}$$

Note that $\max_{j,k\in\{0,\ldots,K\}}\left\|z_{i,j}-z_{i,k}\right\| \le \kappa r(i)/\rho$. Hence, using Lemma A.1 we get

$$\left\|W_iZ_i^{(2)}\right\|^2 < \frac{c_a\kappa^2 r(i)^2}{\rho^2}\,, \qquad (19)$$

where c_a is a constant that depends on the specific algorithm. Combining Eqs. 18 and 19 we obtain

$$\Phi(Z) < N\frac{\overline{e}^{(1)}}{\sigma^2} + N\frac{\kappa^2\overline{e}^{(1)}}{\rho^2} + |N_0|c_a r_{\max}^2\frac{\kappa^2}{\rho^2}.$$

In the specific case of LEM and DFM, a tighter bound can be obtained for $\left\|W_i Z_i^{(2)}\right\|^2$. Note that for LEM and DFM

$$\begin{aligned}\left\|W_i Z_i^{(2)}\right\|^2 &= \sum_{j=1^K} w_{i,j}(z_i^{(2)} - z_{i,j}^{(2)})^2 \\ &\leq \sum_{j=1}^{K} w_{i,j}\frac{\kappa^2}{\rho^2}(x_i^{(2)} - x_{i,j}^{(2)})^2 = \frac{\kappa^2}{\rho^2}e_i^{(1)}.\end{aligned}$$

Combining Eq. 17 and the last inequality we obtain in this case that

$$\Phi(Z) \leq N\frac{\overline{e}^{(1)}}{\sigma^2} + N\frac{\kappa^2\overline{e}^{(1)}}{\rho^2},$$

which completes the proof.

A.7 Lemma A.1

Lemma A.1 *Let $X_i = [x_i, x_{i,1}, \ldots, x_{i,K}]'$ be a local neighborhood. Let $r_i = \max_{j,k}\left\|x_{i,j} - x_{i,k}\right\|$. Then*

$$\|W_i X_i\|_F^2 < c_a r_i^2,$$

where c_a is a constant that depends on the algorithm.

Proof We prove this lemma for each of the different algorithms separately.

- LEM and DFM:

  $$\|W_i X_i\|_F^2 = \sum_{j=1}^{K} w_{i,j}\left\|x_{i,j} - x_i\right\|^2 \leq \left(\sum_{j=1}^{K} w_{i,j}\right) r_i^2 \leq K r_i^2,$$

  where the last inequality holds since $w_{i,j} \leq 1$. Hence $c_a = K$.

- LLE:

  $$\begin{aligned}\|W_i X_i\|_F^2 &= \left\|\sum_{j=1}^{K} w_{i,j}(x_{i,j} - x_i)\right\|^2 \leq \left\|\frac{1}{K}\sum_{j=1}^{K}(x_{i,j} - x_i)\right\|^2 \\ &\leq \frac{1}{K^2}\sum_{j=1}^{K}\left\|x_{i,j} - x_i\right\|^2 \leq \frac{r_i^2}{K},\end{aligned}$$

  where the first inequality holds since $w_{i,j}$ were chosen to minimize $\left\|\sum_{j=1}^{K}\tilde{w}_{i,j}(x_{i,j} - x_i)\right\|^2$. Hence $c_a = 1/K$.

- LTSA:

$$\begin{aligned} \|W_i X_i\|_F^2 &= \left\|(I - P_i P_i')HX_i\right\|_F^2 \leq \left\|(I - P_i P_i')\right\|_F^2 \|HX_i\|_F^2 \\ &\leq K \sum_j \left\|x_{i,j} - \bar{x_i}\right\|^2 \leq K^2 r_i^2 . \end{aligned}$$

  The first equality is just the definition of W_i (see Sec. 2). The matrix $I - P_i P_i'$ is a projection matrix and its square norm is the dimension of its range, which is smaller than $K+1$. Hence $c_a = K^2$.

- HLLE:

$$\|W_i X_i\|_F^2 = \|W_i H X_i\|_F^2 \leq \|W_i\|_F^2 \|HX_i\|_F^2 \leq \frac{d(d+1)}{2}(K+1) r_i^2 .$$

  The first equality holds since $W_i H = W_i(I - \frac{1}{K}\mathbf{1}\mathbf{1}') = W_i$, since the rows of W_i are orthogonal to the vector $\mathbf{1}$ by definition (see Sec. 2). Hence $c_a = \frac{d(d+1)}{2}(K+1)$.

■

A.8 Lemma A.2

Lemma A.2 *Let X be a random variable symmetric around zero with unimodal distribution. Assume that $Var(X) = \sigma^2$. Then $Var(|X|) \geq \frac{\sigma^2}{4}$.*

Proof First note that that the equality holds for $X \sim U(-\sqrt{3}\sigma, \sqrt{3}\sigma)$, where U denotes the uniform distribution. Assume by contradiction that there is a random variable X, symmetric around zero and with unimodal distribution such that $\mathrm{Var}(|X|) < \frac{\sigma^2}{4} - \varepsilon$, where $\varepsilon > 0$. Since $\mathrm{Var}(|X|) = E(|X|^2) - E(|X|)^2$, and $E(|X|^2) = E(X^2) = \mathrm{Var}(X) = \sigma^2$, we have $E(|X|)^2 > \frac{3\sigma^2}{4} + \varepsilon$.

We approximate X by X_n, where X_n is a mixture of uniform random variables, defined as follows. Define $X_n \sim \sum_{i=1}^n p_i^n U(-c_i^n, c_i^n)$ where $p_i^n > 0$, $\sum_{i=1}^n p_i^n = 1$. Note that $E(X_n) = 0$ and that $\mathrm{Var}(X_n) = \sum_{i=1}^n p_i^n (c_i^n)^2/3$. For large enough n, we can choose p_i^n and c_i^n such that $\mathrm{Var}(X_n) = \sigma^2$ and $E(|X - X_n|) < \frac{\varepsilon}{2E(|X|)}$.

Consider the random variable $|X_n|$. Note that using the definitions above we may write $|X_n| = \sum_{i=1}^n p_i^n U(0, c_i^n)$, hence $E(|X_n|) = \frac{1}{2}\sum_{i=1}^n p_i^n c_i^n$. We bound this expression from below. We have

$$\begin{aligned} E(|X_n|)^2 &= E(|X_n - X + X|)^2 \geq (E(|X|) - E(|X_n - X|))^2 \\ &\geq E(|X|)^2 - 2E(|X|)E(|X_n - X|) > \frac{3\sigma^2}{4} . \end{aligned} \tag{20}$$

Let $X_{n-1} = \sum_{i=1}^{n-1} p_i^{n-1} U(-c_i^{n-1}, c_i^{n-1})$ where

$$p_i^{n-1} = \begin{cases} p_i^n & i < n-1 \\ p_{n-1}^n + p_n^n & i = n-1 \end{cases} ,$$

and

$$c_i^{n-1} = \begin{cases} c_i^n & i < n-1 \\ \sqrt{\left((c_{n-1}^n)^2 + (c_n^n)^2\right)} & i = n-1 \end{cases} .$$

Note that $\mathrm{Var}(X_{n-1}) = \sigma^2$ by construction and X_{n-1} is symmetric around zero with unimodal distribution. Using the triangle inequality we obtain

$$E(|X_{n-1}|) = \frac{1}{2}\sum_{i=1}^{n-1} p_i^{n-1} c_i^{n-1} \geq \frac{1}{2}\sum_{i=1}^{n} p_i^n c_i^n = E(|X_n|)\,.$$

Using the same argument recursively, we obtain that $E(|X_1|) \geq E(|X_n|)$. However, $X_1 \sim U(-\sqrt{3}\sigma, \sqrt{3}\sigma)$ and hence $E(|X_1|)^2 = \frac{3\sigma^2}{4}$. Since by Eq. 20, $E(|X_n|)^2 > \frac{3\sigma^2}{4}$ we have a contradiction. ■

References

M. Balasubramanian, E. L. Schwartz, J. B. Tenenbaum, V. de Silva, and J. C. Langford. The isomap algorithm and topological stability. *Science*, 295(5552):7, 2002.

M. Belkin and P. Niyogi. Towards a theoretical foundation for Laplacian-based manifold methods. In *COLT*, pages 486–500, 2005.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comp.*, 15(6):1373–1396, 2003.

M. Bernstein, V. de Silva, J. C. Langford, and J. B. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Stanford University, Stanford, Available at http://isomap.stanford.edu, 2000.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21 (1):5–30, 2006.

V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 721–728. MIT Press, 2003.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci. U.S.A.*, 100(10):5591–5596, 2004.

S. Gerber, T. Tasdizen, and R. Whitaker. Robust non-linear dimensionality reduction using successive 1-dimensional Laplacian eigenmaps. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 281–288. Omnipress, 2007.

Y. Goldberg, A. Zakai, and Y. Ritov. Does the Laplacian Eigenmap algorithm work? Unpublished, May, 2007.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 1983.

J. Hamm, D. Lee, and L. K. Saul. Semisupervised alignment of manifolds. In Robert G. Cowell and Zoubin Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 120–127, 2005.

M. Hein, J. Y. Audibert, and U. von Luxburg. From graphs to manifolds - weak and strong pointwise consistency of graph Laplacians. In *COLT*, pages 470–485, 2005.

X. Huo and A. K. Smith. Performance analysis of a manifold learning algorithm in dimension reduction. Technical Paper, Statistics in Georgia Tech, Georgia Institute of Technology, March 2006.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *J. Mach. Learn. Res.*, 4:119–155, 2003.

F. Sha and L. K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML)*, pages 784–791, 2005.

A. Singer. From graph to manifold Laplacian: the convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):135–144, 2006.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

T. Wittman. MANIfold learning matlab demo. `http://www.math.umn.edu/~wittman/mani/`, retrieved Jan. 2007.

Z. Y. Zhang and H. Y. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Sci. Comp*, 26(1):313–338, 2004.

Journal of Machine Learning Research 9 (2008) 1941-1979 Submitted 6/07; Revised 2/08; Published 9/08

Complete Identification Methods for the Causal Hierarchy

Ilya Shpitser ILYAS@CS.UCLA.EDU
Judea Pearl JUDEA@CS.UCLA.EDU
Cognitive Systems Laboratory
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095, USA

Editor: Peter Spirtes

Abstract

We consider a hierarchy of queries about causal relationships in graphical models, where each level in the hierarchy requires more detailed information than the one below. The hierarchy consists of three levels: associative relationships, derived from a joint distribution over the observable variables; cause-effect relationships, derived from distributions resulting from external interventions; and counterfactuals, derived from distributions that span multiple "parallel worlds" and resulting from simultaneous, possibly conflicting observations and interventions. We completely characterize cases where a given causal query can be computed from information lower in the hierarchy, and provide algorithms that accomplish this computation. Specifically, we show when effects of interventions can be computed from observational studies, and when probabilities of counterfactuals can be computed from experimental studies. We also provide a graphical characterization of those queries which cannot be computed (by any method) from queries at a lower layer of the hierarchy.

Keywords: causality, graphical causal models, identification

1. Introduction

The human mind sees the world in terms of causes and effects. Understanding and mastering our environment hinges on answering questions about cause-effect relationships. In this paper we consider three distinct classes of causal questions forming a hierarchy.

The first class of questions involves associative relationships in domains with uncertainty, for example, "I took an aspirin after dinner, will I wake up with a headache?" The tools needed to formalize and answer such questions are the subject of probability theory and statistics, for they require computing or estimating some aspects of a joint probability distribution. In our aspirin example, this requires estimating the conditional probability $P(headache|aspirin)$ in a population that resembles the subject in question, that is, sharing age, sex, eating habits and any other traits that can be measured. Associational relationships, as is well known, are insufficient for establishing causation. We nevertheless place associative questions at the base of our causal hierarchy, because the probabilistic tools developed in studying such questions are instrumental for computing more informative causal queries, and serve therefore as an easily available starting point from which such computations can begin.

The second class of questions involves responses of outcomes of interest to outside interventions, for instance, "if I take an aspirin now, will I wake up with a headache?" Questions of this type are normally referred to as *causal effects*, sometimes written as $P(headache|do(aspirin))$. They

differ, of course from the associational counterpart
$P(headache|aspirin)$, because all mechanisms which normally determine aspirin taking behavior, for example, taste of aspirin, family advice, time pressure, etc. are irrelevant in evaluating the effect of a new decision.

To estimate effects, scientists normally perform randomized experiments where a sample of units drawn from the population of interest is subjected to the specified manipulation directly. In our aspirin example, this might involve treating a group of subjects with aspirin and comparing their response to untreated subjects, both groups being selected at random from a population resembling the decision maker in question. In many cases, however, such a direct approach is not possible due to expense or ethical considerations. Instead, investigators have to rely on observational studies to infer effects. A fundamental question in causal analysis is to determine when effects can be inferred from statistical information, encoded as a joint probability distribution, obtained under normal, intervention-free behavior. A key point here is that in order to make causal inferences from statistics, additional causal assumptions are needed. This is because without any assumptions it is possible to construct multiple "causal stories" which can disagree wildly on what effect a given intervention can have, but agree precisely on all observables. For instance, smoking may be highly correlated with lung cancer either because it causes lung cancer, or because people who are genetically predisposed to smoke may also have a gene responsible for a higher cancer incidence rate. In the latter case there will be no effect of smoking on cancer. Distinguishing between such causal stories requires additional, non-statistical language. In this paper, the language that we use for this purpose is the language of graphs, and our causal assumptions will be encoded by a special directed graph called a *causal diagram*.

The use of directed graphs to represent causality is a natural idea that arose multiple times independently: in genetics (Wright, 1921), econometrics (Haavelmo, 1943), and artificial intelligence (Pearl, 1988; Spirtes et al., 1993; Pearl, 2000). A causal diagram encodes variables of interest as nodes, and possible direct causal influences between two variables as arrows. Associated with each node in a causal diagram is a stable causal mechanism which determines its value in terms of the values of its parents. Unlike Bayesian networks (Pearl, 1988), the relationships between variables are assumed to be deterministic and uncertainty arises due to the presence of unobserved variables which have influence on our domain.

The first question we consider is under what conditions the effect of a given intervention can be computed from just the joint distribution over observable variables, which is obtainable by statistical means, and the causal diagram, which is either provided by a human expert, or inferred from experimental studies. This *identification problem* has received consideration attention in the statistics, epidemiology, and causal inference communities (Pearl, 1993a; Spirtes et al., 1993; Pearl and Robins, 1995; Pearl, 1995; Kuroki and Miyakawa, 1999; Pearl, 2000). In the subsequent sections, we solve the identification problem for causal effects by providing a graphical characterization for all non-identifiable effects, and an algorithm for computing all identifiable effects. Note that this identification problem actually involves two "worlds:" the original world where no interventions took place furnishes us with a probability distribution from which to make inferences about the second, post-intervention world. The crucial feature of causal effect queries which distinguishes them from more complex questions in our hierarchy is that they are restricted to the post-intervention world alone.

The third and final class of queries we consider are *counterfactual* or "what-if" questions which arise when we simultaneously ask about multiple hypothetical worlds, with potentially conflicting

interventions or observations. An example of such a question would be "I took an aspirin, and my headache is gone; would I have had a headache had I not taken that aspirin?" Unlike questions involving interventions, counterfactuals contain conflicting information: in one world aspirin was taken, in another it was not. It is unclear therefore how to set up an effective experimental procedure for evaluating counterfactuals, let alone how to compute counterfactuals from observations alone. If everything about our causal domain is known, in other words if we have knowledge of both the causal mechanisms and the distributions over unobservable variables, it is possible to compute counterfactual questions directly (Balke and Pearl, 1994b). However, knowledge of precise causal mechanisms is not generally available, and the very nature of unobserved variables means their stochastic behavior cannot be estimated directly. We therefore consider the more practical question of how to compute counterfactual questions from both experimental studies and the structure of the causal diagram.

It may seem strange, in light of what we said earlier about the difficulty of conducting experimental studies, that we take such studies as given. It is nevertheless important that we understand when it is that "what-if" questions involving multiple worlds can be inferred from quantities computable in one world. Our hierarchical approach to identification allows us to cleanly separate difficulties that arise due to multiplicity of worlds from those involved in the identification of causal effects. We provide a complete solution to this version of the identification problem by giving algorithms which compute identifiable counterfactuals from experimental studies, and provide graphical conditions for the class of non-identifiable counterfactuals, where our algorithms fail. Our results can, of course, be combined to give conditions where counterfactuals can be computed from observational studies.

The paper is organized as follows. Section 2 introduces the notation and mathematical machinery needed for causal analysis. Section 3 considers the problem of identifying causal effects from observational studies. Section 4 considers identification of counterfactual queries, while Section 5 summarizes the conclusions. Most of the proofs are deferred to the appendix. This paper consolidates and expands previous results (Shpitser and Pearl, 2006a,b, 2007). Some of the results found in this paper were also derived independently elsewhere (Huang and Valtorta, 2006b,a).

2. Notation and Definitions

The primary object of causal inquiry is a probabilistic causal model. We will denote variables by uppercase letters, and their values by lowercase letters. Similarly, sets of variables will be denoted by bold uppercase, and sets of values by bold lowercase.

Definition 1 *A probabilistic causal model (PCM) is a tuple* $M = \langle \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{F}, P(\boldsymbol{u}) \rangle$*, where*

- $\boldsymbol{U}$ *is a set of background or exogenous variables, which cannot be observed or experimented on, but which affect the rest of the model.*
- $\boldsymbol{V}$ *is a set* $\{V_1, ..., V_n\}$ *of observable or endogenous variables. These variables are functionally dependent on some subset of* $\boldsymbol{U} \cup \boldsymbol{V}$*.*
- $\boldsymbol{F}$ *is a set of functions* $\{f_1, ..., f_n\}$ *such that each* f_i *is a mapping from a subset of* $\boldsymbol{U} \cup \boldsymbol{V} \setminus \{V_i\}$ *to* V_i*, and such that* $\bigcup \boldsymbol{F}$ *is a function from* $\boldsymbol{U}$ *to* $\boldsymbol{V}$*.*
- $P(\boldsymbol{u})$ *is a joint probability distribution over* $\boldsymbol{U}$*.*

The set of functions **F** in this definition corresponds to the causal mechanisms, while **U** represents the background context that influences the observable domain of discourse **V**, yet remains outside it. Our ignorance of the background context is represented by a distribution $P(\mathbf{u})$. This distribution, together with the mechanisms in **F**, induces a distribution $P(\mathbf{v})$ over the observable domain. The causal diagram, our vehicle for expressing causal assumptions, is defined by the causal model as follows. Each observable variable $V_i \in \mathbf{V}$ corresponds to a vertex in the graph. Any two variables $V_i \in \mathbf{U} \cup \mathbf{V}$, $V_j \in \mathbf{V}$ such that V_i appears in the description of f_j are connected by a directed arrow from V_i to V_j. Furthermore, we make two additional assumptions in this paper. The first is that $P(\mathbf{u}) = \prod_{u_i \in \mathbf{u}} P(u_i)$, and each $U_i \in \mathbf{U}$ is used in at most two functions in F.[1] The second is that all induced graphs must be acyclic. Models in which these two assumptions hold are called recursive semi-Markovian. A graph defined as above from a causal model M is said to be a causal diagram *induced* by M. Graphs induced by semi-Markovian models are themselves called semi-Markovian. Fig. 1 and Fig. 2 show some examples of causal diagrams of recursive semi-Markovian models.

The functions in **F** are assumed to be *modular* in a sense that changes to one function do not affect any other. This assumption allows us to model how a PCM would react to changes imposed from the outside. The simplest change that is possible for causal mechanisms of a variable set **X** would be one that removes the mechanisms entirely and sets **X** to a specific value **x**. This change, denoted by $do(\mathbf{x})$ (Pearl, 2000), is called an *intervention*. An intervention $do(\mathbf{x})$ applied to a model M results in a *submodel* $M_{\mathbf{X}}$. The effects of interventions will be formulated in several ways. For any given **u**, the effect of $do(\mathbf{x})$ on a set of variables **Y** will be represented by *counterfactual variables* $Y_{\mathbf{X}}(\mathbf{u})$, where $Y \in \mathbf{Y}$. As **U** varies, the counterfactuals $Y_{\mathbf{X}}(\mathbf{u})$ will vary as well, and their *interventional distribution*, denoted by $P(\mathbf{y}|do(\mathbf{x}))$ or $P_{\mathbf{x}}(\mathbf{y})$ will be used to define the effect of **x** on **Y**. We will denote the event "variable Y attains value y in $M_{\mathbf{X}}$" by the shorthand $y_{\mathbf{X}}$.

Interventional distributions are a mathematical formalization of an intuitive notion of effect of action. We now define joint probabilities on counterfactuals, in multiple worlds, which will serve as the formalization of counterfactual queries. Consider a conjunction of events $\gamma = y^1_{\mathbf{X}^1} \wedge ... \wedge y^k_{\mathbf{X}^k}$. If all the subscripts $\mathbf{x}^i$ are the same and equal to **x**, γ is simply a set of assignments of values to variables in $M_{\mathbf{X}}$, and $P(\gamma) = P_{\mathbf{X}}(y^1, ..., y^k)$. However, if the actions $do(\mathbf{x}^i)$ are not the same, and potentially contradictory, a single submodel is no longer sufficient. Instead, γ is really invoking multiple causal worlds, each represented by a submodel $M_{\mathbf{x}^i}$. We assume each submodel shares the same set of exogenous variables **U**, corresponding to the shared causal context or background history of the hypothetical worlds. Because the submodels are linked by common context, they can really be considered as one large causal model, with its own induced graph, and joint distribution over observable variables. $P(\gamma)$ can then be defined as a marginal distribution in this causal model. Formally, $P(\gamma) = \sum_{\{\mathbf{u}|\mathbf{u} \models \gamma\}} P(\mathbf{u})$, where $\mathbf{u} \models \gamma$ is taken to mean that each variable assignment in γ holds true in the corresponding submodel of M when the exogenous variables **U** assume values **u**. In this way, $P(\mathbf{u})$ induces a distribution on all possible counterfactual variables in M. In this paper, we will represent counterfactual utterances by joint distributions such as $P(\gamma)$ or conditional distributions such as $P(\gamma|\delta)$, where γ and δ are conjunctions of counterfactual events. Pearl (2000) discusses counterfactuals, and their probabilistic representation used in this paper in greater depth.

1. Our results are generalizable to other $P(\mathbf{u})$ distributions which may not have such a simple form, but which can be represented by a set of bidirected arcs in such a way that whenever two sets of **U** variables are d-separated from each other, they are marginally independent. However, the exact conditions under which this graphical representation is valid are beyond the scope of this paper.

A fundamental question in causal inference is whether a given causal question, either interventional or counterfactual in nature, can be uniquely specified by the assumptions embodied in the causal diagram, and easily available information, usually statistical, associated with the causal model. To get a handle on this question, we introduce an important notion of *identifiability* (Pearl, 2000).

Definition 2 (identifiability) *Consider a class of models $\mathbf{M}$ with a description T, and objects ϕ and θ computable from each model. We say that ϕ is θ-identified in T if ϕ is uniquely computable from θ in any $M \in \mathbf{M}$. In this case all models in $\mathbf{M}$ which agree on θ will also agree on ϕ.*

If ϕ is θ-identifiable in T, we write $T, \theta \vdash_{id} \phi$. Otherwise, we write $T, \theta \nvdash_{id} \phi$. The above definition leads immediately to the following corollary which we will use to prove non-identifiability results.

Corollary 3 *Let T be a description of a class of models $\mathbf{M}$. Assume there exist $M^1, M^2 \in \mathbf{M}$ that share objects θ, while ϕ in M^1 is different from ϕ in M^2. Then $T, \theta \nvdash_{id} \phi$.*

In our context, the objects ϕ, θ are probability distributions derived from the PCM, where θ represents available information, while ϕ represents the quantity of interest. The description T is a specification of the properties shared all causal models under consideration, or, in other words, the set of assumptions we wish to impose on those models. Since we chose causal graphs as a language for specifying assumptions, T corresponds to a given graph.

Graphs earn their ubiquity as a specification language because they reflect in many ways the way people store experiential knowledge, especially cause-effect relationships. The ease with which people embrace graphical metaphors for causal and probabilistic notions—ancestry, neighborhood, flow, and so on—are proof of this affinity, and help ensure that the assumptions specified are meaningful and reliable. A consequence of this is that probabilistic dependencies among variables can be verified by checking if the flow of influence is blocked along paths linking the variables. By a path we mean a sequence of distinct nodes where each node is connected to the next in the sequence by an edge. The precise way in which the flow of dependence can be blocked is defined by the notion of d-separation (Pearl, 1986; Verma, 1986; Pearl, 1988). Here we generalize d-separation somewhat to account for the presence of bidirected arcs in causal diagrams.

Definition 4 (d-separation) *A path p in G is said to be d-separated by a set $\mathbf{Z}$ if and only if either*

1. *p contains one of the following three patterns of edges: $I \rightarrow M \rightarrow J$, $I \leftrightarrow M \rightarrow J$, or $I \leftarrow M \rightarrow J$, such that $M \in \mathbf{Z}$, or*

2. *p contains one of the following three patterns of edges: $I \rightarrow M \leftarrow J$, $I \leftrightarrow M \leftarrow J$, $I \leftrightarrow M \leftrightarrow J$, such that $De(M)_G \cap \mathbf{Z} = \emptyset$.*

Two sets $\mathbf{X}, \mathbf{Y}$ are said to be d-separated given $\mathbf{Z}$ in G if all paths from $\mathbf{X}$ to $\mathbf{Y}$ in G are d-separated by $\mathbf{Z}$. Paths or sets which are not d-separated are said to be d-connected. What allows us to connect this notion of blocking of paths in a causal diagram to the notion of probabilistic independence among variables is that the probability distribution over $\mathbf{V}$ and $\mathbf{U}$ in a causal model can be represented as a product of factors, such that each observable node has a factor corresponding

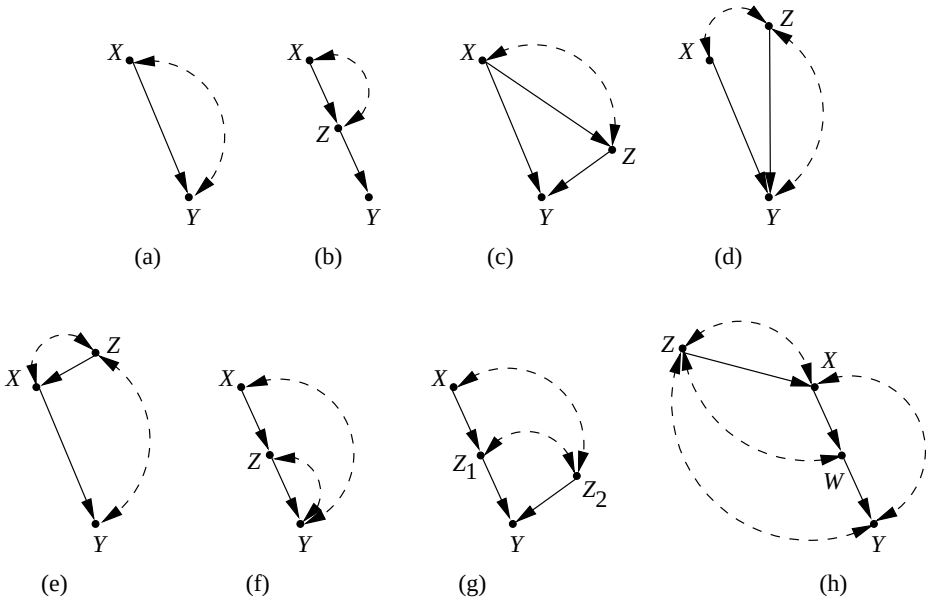


Figure 1: Causal graphs where $P(y|do(\mathbf{x}))$ is not identifiable

to its conditional distribution given the values of its parents in the graph. In other words, $P(\mathbf{v},\mathbf{u}) = \prod_i P(x_i|pa(x_i)_G)$.

Whenever the above factor decomposition holds for a distribution $P(\mathbf{v},\mathbf{u})$ and a graph G, we say G is an I-map of $P(\mathbf{v},\mathbf{u})$. The following theorem links d-separation of vertex sets in an I-map G with the independence of corresponding variable sets in P.

Theorem 5 *If sets $\boldsymbol{X}$ and $\boldsymbol{Y}$ are d-separated by $\boldsymbol{Z}$ in G, then $\boldsymbol{X}$ is independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in every P for which G is an I-map. Furthermore, the causal diagram induced by any semi-Markovian PCM M is an I-map of the distribution $P(\boldsymbol{v},\boldsymbol{u})$ induced by M.*

Note that it's easy to rephrase the above theorem in terms of ordinary directed acyclic graphs, since each semi-Markovian graph is really an abbreviation where each bidirected arc stands for two directed arcs emanating from a hidden common cause. We will abbreviate this statement of d-separation, and corresponding independence by $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z})_G$, following the notation of Dawid (1979). For example in the graph shown in Fig. 6 (a), $X \not\perp\!\!\!\perp Y$ and $X \perp\!\!\!\perp Y|Z$, while in Fig. 6 (b), $X \perp\!\!\!\perp Y$ and $X \not\perp\!\!\!\perp Y|Z$.

Finally we consider the axioms and inference rules we will need. Since PCMs contain probability distributions, the inference rules we would use to compute queries in PCMs would certainly include the standard axioms of probability. They also include a set of axioms which govern the behavior of counterfactuals, such as Effectiveness, Composition, etc. (Galles and Pearl, 1998; Halpern, 2000; Pearl, 2000). However, in this paper, we will concentrate on a set of three identities applicable to interventional distributions known as do-calculus (Pearl, 1993b, 2000):

- Rule 1: $P_{\mathbf{X}}(\mathbf{y}|\mathbf{z},\mathbf{w}) = P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$ if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}}}$
- Rule 2: $P_{\mathbf{X},\mathbf{Z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{X}}(\mathbf{y}|\mathbf{z},\mathbf{w})$ if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}}\underline{\mathbf{Z}}}}$

- Rule 3: $P_{\mathbf{X},\mathbf{Z}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$ if $(\mathbf{Y} \perp\!\!\!\perp \mathbf{Z}|\mathbf{X},\mathbf{W})_{G_{\overline{\mathbf{X}},\underline{Z(\mathbf{W})}}}$

where $Z(\mathbf{W}) = \mathbf{Z} \setminus An(\mathbf{W})_{G_{\overline{\mathbf{X}}}}$, and $G_{\overline{\mathbf{X}},\underline{\mathbf{Y}}}$ stands for a directed graph obtained from G by removing all incoming arrows to $\mathbf{X}$ and all outgoing arrows from $\mathbf{Y}$. The rules of do-calculus provide a way of linking ordinary statistical distributions with distributions resulting from various manipulations.

In the remainder of this section we will introduce relevant graphs and graph-theoretic terminology which we will use in the rest of the paper. First, having defined causal diagrams induced by natural causal models, we consider the graphs induced by models derived from interventional and counterfactual queries. We note that in a given submodel $M_{\mathbf{X}}$, the mechanisms determining $\mathbf{X}$ no longer make use of the parents of $\mathbf{X}$ to determine their values, but instead set them independently to constant values $\mathbf{x}$. This means that the induced graph of $M_{\mathbf{X}}$ derived from a model M inducing graph G can be obtained from G by removing all arrows incoming to $\mathbf{X}$, in other words $M_{\mathbf{X}}$ induces $G_{\overline{\mathbf{X}}}$. A counterfactual $\gamma = y^1_{\mathbf{x}^1} \wedge \ldots \wedge y^k_{\mathbf{x}^k}$, as we already discussed invokes multiple hypothetical causal worlds, each represented by a submodel, where all worlds share the same background context $\mathbf{U}$. A naive way to graphically represent these worlds would be to consider all the graphs $G_{\overline{\mathbf{X}^i}}$ and have them share the $\mathbf{U}$ nodes. It turns out this representation suffers from certain problems. In Section 4 we discuss this issue in more detail and suggest a more appropriate graphical representation of counterfactual situations.

We denote $Pa(.)_G, Ch(.)_G, An(.)_G, De(.)_G$ as the sets of parents, children, ancestors, and descendants of a given set in G. We denote $G_{\mathbf{X}}$ to be the subgraph of G containing all vertices in $\mathbf{X}$, and edges between these vertices, while the set of vertices in a given graph G is given by $\mathbf{ver}(G)$. As a shorthand, we denote $G_{\mathbf{ver}(G)\setminus\mathbf{ver}(G')}$ as $G \setminus G'$ or $G \setminus \mathbf{X}$, if $\mathbf{X} = \mathbf{ver}(G')$, and G' is a subgraph of G. We will call the set $\{X \in G | De(X)_G = \emptyset\}$ the *root set* of G. A path connecting X and Y which begins with an arrow pointing to X is called a *back-door path* from X, while a path beginning with an arrow pointing away from X is called a *front-door path* from X.

The goal of this paper is a complete characterization of causal graphs which permit the answering of causal queries of a given type. This characterization requires the introduction of certain key graph structures.

Definition 6 (tree) *A graph G such that each vertex has at most one child, and only one vertex (called the root) has no children is called a tree.*

Note that this definition reverses the usual direction of arrows in trees as they are generally understood in graph theory. If we ignore bidirected arcs, graphs in Fig. 1 (a), (b), (d), (e), (f), (g), and (h) are trees.

Definition 7 (forest) *A graph G such that each vertex has at most one child is called a forest.*

Note that the above two definitions reverse the arrow directionality usual for these structures.

Definition 8 (confounded path) *A path where all directed arrowheads point at observable nodes, and never away from observable nodes is called a confounded path.*

The graph in Fig. 1 (g) contains a confounded path from Z_1 to Z_2.

Definition 9 (c-component) *A graph G where any pair of observable nodes is connected by a confounded path is called a c-component (confounded component).*

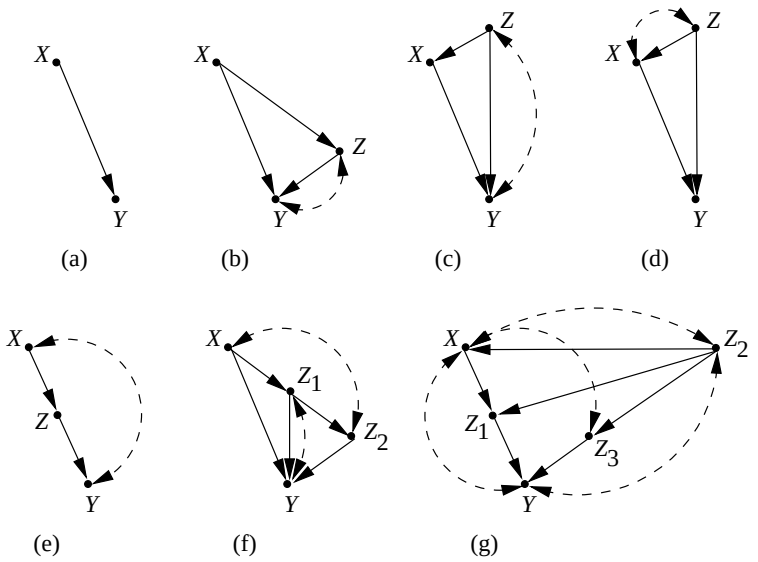


Figure 2: Causal graphs admitting identifiable effect $P(y|do(x))$

Graphs in Fig. 1 (a), (d), (e), (f), and (h) are c-components. Some graphs contain multiple c-components, for example the graph in Fig. 1 (b) has two maximal c-components: $\{Y\}$, and $\{X,Z\}$. We will denote the set of maximal c-components of a given graph G by $C(G)$. The importance of c-components stems from the fact that that the observational distribution $P(\mathbf{v})$ can be expressed as a product of factors $P_{\mathbf{V}\setminus\mathbf{S}}(\mathbf{s})$, where each $\mathbf{s}$ is a set of nodes forming a c-component. This important property is known as *c-component factorization*, and we will this property extensively in the remainder of the manuscript to decompose identification problems into smaller subproblems.

In the following sections, we will show how the graph structures we defined in this section are key for characterizing cases when $P_{\mathbf{X}}(\mathbf{y})$ and $P(\gamma)$ can be identified from available information.

3. Identification of Causal Effects

Like probabilistic dependence, the notion of causal effect of X on Y has an interpretation in terms of flow. Intuitively, X has an effect on Y if changing X causes Y to change. Since intervening on X cuts off X from the normal causal influences of its parents in the graph, we can interpret the causal effect of X on Y as the flow of dependence which leaves X via outgoing arrows only.

Recall that our ultimate goal is to express distributions of the form $P(\mathbf{y}|do(\mathbf{x}))$ in terms of the joint distribution $P(\mathbf{v})$. The interpretation of effect as downward dependence immediately suggests a set of graphs where this is possible. Specifically, whenever all d-connected paths from $\mathbf{X}$ to $\mathbf{Y}$ are front-door from $\mathbf{X}$, the causal effect $P(\mathbf{y}|do(\mathbf{x}))$ is equal to $P(\mathbf{y}|\mathbf{x})$. In graphs shown in Fig. 2 (a) and (b) causal effect $P(y|do(x))$ has this property.

In general, we don't expect acting on $\mathbf{X}$ to produce the same effect as observing $\mathbf{X}$ due to the presence of back-door paths between $\mathbf{X}$ and $\mathbf{Y}$. However, d-separation gives us a way to block undesirable paths by conditioning. If we can find a set $\mathbf{Z}$ that blocks all back-door paths from $\mathbf{X}$ to $\mathbf{Y}$, we obtain the following: $P(\mathbf{y}|do(\mathbf{x})) = \sum_{\mathbf{Z}} P(\mathbf{y}|\mathbf{z},do(\mathbf{x}))P(\mathbf{z}|do(\mathbf{x}))$. The term $P(\mathbf{y}|\mathbf{z},do(\mathbf{x}))$ is reduced to $P(\mathbf{y}|\mathbf{z},\mathbf{x})$ since the influence flow from $\mathbf{X}$ to $\mathbf{Y}$ is blocked by $\mathbf{Z}$. However, the act of

adjusting for $\mathbf{Z}$ introduced a new effect we must compute, corresponding to the term $P(\mathbf{z}|do(\mathbf{x}))$. If it so happens that no variable in $\mathbf{Z}$ is a descendant of $\mathbf{X}$, we can reduce this term to $P(\mathbf{z})$ using the intuitive argument that acting on effects should not influence causes, or a more formal appeal to rule 3 of do-calculus. Computing effects in this way is always possible if we can find a set $\mathbf{Z}$ blocking all back-door paths which contains no descendants of $\mathbf{X}$. This is known as the *back-door criterion* (Pearl, 1993a, 2000). Figs. 2 (c) and (d) show some graphs where the node z satisfies the back-door criterion with respect to $P(y|do(x))$, which means $P(y|do(x))$ is identifiable.

The back-door criterion can fail—a common way involves a confounder that is unobserved, which prevents adjusting for it. Surprisingly, it is sometimes possible to identify the effect of $\mathbf{X}$ on $\mathbf{Y}$ even in the presence of such a confounder. To do so, we want to find a set $\mathbf{Z}$ located downstream of $\mathbf{X}$ but upstream of $\mathbf{Y}$, such that the downward flow of the effect of $\mathbf{X}$ on $\mathbf{Y}$ can be decomposed into the flow from $\mathbf{X}$ to $\mathbf{Z}$, and the flow from $\mathbf{Z}$ to $\mathbf{Y}$. Clearly, in order for this to happen $\mathbf{Z}$ must d-separate all front-door paths from $\mathbf{X}$ to $\mathbf{Y}$. However, in order to make sure that the component effects $P(\mathbf{z}|do(\mathbf{x}))$ and $P(\mathbf{y}|do(\mathbf{z}))$ are themselves identifiable, and combine appropriately to form $P(\mathbf{y}|do(\mathbf{x}))$, we need two additional assumptions: there are no back-door paths from $\mathbf{X}$ to $\mathbf{Z}$, and all back-door paths from $\mathbf{Z}$ to $\mathbf{Y}$ are blocked by $\mathbf{X}$. It turns out that these three conditions imply that $P(\mathbf{y}|do(\mathbf{x})) = \sum_{\mathbf{Z}} P(\mathbf{y}|do(\mathbf{z}))P(\mathbf{z}|do(\mathbf{x}))$, and the latter two conditions further imply that the first term is identifiable by the back-door criterion and equal to $\sum_{\mathbf{Z}} P(\mathbf{y}|\mathbf{z},\mathbf{x})P(\mathbf{x})$, while the second term is equal to $P(\mathbf{z}|\mathbf{x})$. Whenever these three conditions hold, the effect of $\mathbf{X}$ on $\mathbf{Y}$ is identifiable. This is known as the *front-door criterion* (Pearl, 1995, 2000). The front-door criterion holds in the graph shown in Fig. 2 (e).

Unfortunately, in some graphs neither the front-door, nor the back-door criterion holds. The simplest such graph, known as the bow arc graph due to its shape, is shown in Fig. 1 (a). The back-door criterion fails since the confounder node is unobservable, while the front-door criterion fails since no intermediate variables between X and Y exist in the graph. While the failure of these two criteria does not imply non-identification, in fact the effect $P(y|do(x))$ is identifiable in Fig. 2 (f), (g) despite this failure, a simple argument shows that $P(y|do(x))$ is not identifiable in the bow arc graph.

Theorem 10 *$P(\boldsymbol{v}), G \not\vdash_{id} P(y|do(x))$ in G shown in Fig. 1 (a).*

Since we are interested in completely characterizing graphs where a given causal effect $P(\mathbf{y}|do(\mathbf{x}))$ is identifiable, it would be desirable to list difficult graphs like the bow arc graph which prevent identification of causal effects, in the hope of eventually making such a list complete and finding a way to identify effects in all graphs not on the list. We start constructing this list by considering graphs which generalize the bow arc graph since they can contain more than two nodes, but which also inherit its difficult structure. We call such graphs C-trees.

Definition 11 (C-tree) *A graph G which is both a C-component and a tree is called a C-tree.*

We call a C-tree with a root node Y Y-rooted. The graphs in Fig. 1 (a), (d), (e), (f), and (h) are Y-rooted C-trees. It turns out that in any Y-rooted C-tree, the effect of any subset of nodes, other than Y, on the root Y is not identifiable.

Theorem 12 *Let G be a Y-rooted C-tree. Let $\boldsymbol{X}$ be any subset of observable nodes in G which does not contain Y. Then $P(\boldsymbol{v}), G \not\vdash_{id} P(y|do(\boldsymbol{x}))$.*

C-trees play a prominent role in the identification of *direct effects*. Intuitively, the direct effect of X on Y exists if there is an arrow from X to Y in the graph, and corresponds to the flow of influence along this arrow. However, simply considering changes in Y after fixing X is insufficient for isolating direct effect, since X can influence Y along other, longer front-door paths than the direct arrow. In order to disregard such influences, we also fix all other parents of Y (which as noted earlier removes all arrows incoming to these parents and thus to Y). The expression corresponding to the direct effect of X on Y is then $P(y|do(pa(y)))$. The following theorem links C-trees and direct effects.

Theorem 13 *$P(\boldsymbol{v}), G \not\vdash_{id} P(y|do(pa(y)))$ if and only if there exists a subgraph of G which is a Y-rooted C-tree.*

This theorem might suggest that C-trees might play an equally strong role in identifying arbitrary effects on a single variable, not just direct effects. Unfortunately, this turns out not to be the case, due to the following lemma.

Lemma 14 (downward extension lemma) *Let $\boldsymbol{V}$ be the set of observable nodes in G, and $P(\boldsymbol{v})$ the observable distribution of models inducing G. Assume $P(\boldsymbol{v}), G \not\vdash_{id} P(\boldsymbol{y}|do(\boldsymbol{x}))$. Let G' contain all the nodes and edges of G, and an additional node Z which is a child of all nodes in $\boldsymbol{Y}$. Then if $P(\boldsymbol{v}, z)$ is the observable distribution of models inducing G', then $P(\boldsymbol{v}, z), G' \not\vdash_{id} P(z|do(\boldsymbol{x}))$.*

Proof Let $|Z| = \prod_{Y_i \in \mathbf{Y}} |Y_i| = n$. By construction, $P(z|do(\mathbf{x})) = \sum_{\mathbf{y}} P(z|\mathbf{y})P(\mathbf{y}|do(\mathbf{x}))$. Due to the way we set the arity of Z, $P(Z|\mathbf{Y})$ is an n by n matrix which acts as a linear map which transforms $P(\mathbf{y}|do(\mathbf{x}))$ into $P(z|do(\mathbf{x}))$. Since we can arrange this linear map to be one to one, any proof of non-identifiability of $P(\mathbf{y}|do(\mathbf{x}))$ immediately extends to the proof of non-identifiability of $P(z|do(\mathbf{x}))$. ■

What this lemma shows is that identification of effects on a singleton is not any simpler than the general problem of identification of effect on a set. To find difficult graphs which prevent identification of effects on sets, we consider a multi-root generalization of C-trees.

Definition 15 (c-forest) *A graph G which is both a C-component and a forest is called a C-forest.*

If a given C-forest has a set of root nodes $\mathbf{R}$, we call it $\mathbf{R}$-rooted. Graphs in Fig. 3 (a), (b) are $\{Y1, Y2\}$-rooted C-forests. A naive way to generalize Theorem 12 would be to state that if G is an $\mathbf{R}$-rooted C-forest, then the effect of any set $\mathbf{X}$ that does not intersect $\mathbf{R}$ is not identifiable. However, as we later show, this is not true. Specifically, we later prove that $P(y1, y2|do(x))$ in the graph in Fig. 3 (a) is identifiable. To formulate the correct generalization of Theorem 12, we must understand what made C-trees difficult for the purposes of identifying effects on the root Y. It turned out that for particular function choices, the effects of ancestors of Y on Y precisely cancelled themselves out so even though Y itself was dependent on its parents, it was observationally indistinguishable from a constant function. To get the same canceling of effects with C-forests, we must define a more complex graphical structure.

Definition 16 (hedge) *Let $\boldsymbol{X}, \boldsymbol{Y}$ be sets of variables in G. Let F, F' be $\boldsymbol{R}$-rooted C-forests in G such that F' is a subgraph of F, $\boldsymbol{X}$ only occur in F, and $\boldsymbol{R} \in An(\boldsymbol{Y})_{G_{\overline{\boldsymbol{X}}}}$. Then F and F' form a hedge for $P(\boldsymbol{y}|do(\boldsymbol{x}))$.*

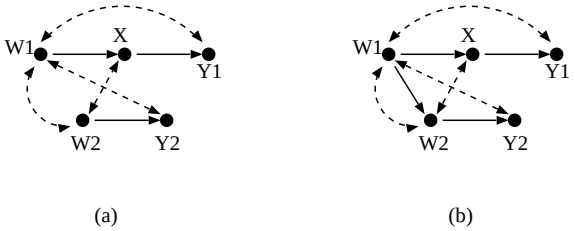


Figure 3: (a) A graph hedge-less for $P(y|do(x))$. (b) A graph containing a hedge for $P(y|do(x))$.

The graph in Fig. 3 (b) contains a hedge for $P(y1,y2|do(x))$. The mental picture for a hedge is as follows. We start with a C-forest F'. Then, F' grows new branches, while retaining the same root set, and becomes F. Finally, we "trim the hedge," by performing the action $do(\mathbf{x})$ which has the effect of removing some incoming arrows in $F \setminus F'$ (the subgraph of F consisting of vertices not a part of F'). Note that any Y-rooted C-tree and its root node Y form a hedge. The right generalization of Theorem 12 can be stated on hedges.

Theorem 17 *Let F, F' be subgraphs of G which form a hedge for $P(\boldsymbol{y}|do(\boldsymbol{x}))$. Then $P(\boldsymbol{v}), G \not\vdash_{id} P(\boldsymbol{y}|do(\boldsymbol{x}))$.*

Proof outline As before, assume binary variables. We let the causal mechanisms of one of the models consists entirely of bit parity functions. The second model also computes bit parity for every mechanism, except those nodes in F' which have parents in F ignore the values of those parents. It turns out that these two models are observationally indistinguishable. Furthermore, any intervention in $F \setminus F'$ will break the bit parity circuits of the models. This break will be felt at the root set **R** of the first model, but not of the second, by construction. ■

Unlike the bow arc graph, and C-trees, hedges prevent identification of effects on multiple variables at once. Certainly a complete list of all possible difficult graphs must contain structures like hedges. But are there other kinds of structures that present problems? It turns out that the answer is "no," any time an effect is not identifiable in a causal model (if we make no restrictions on the type of function that can appear), there is a hedge structure involved. To prove that this is so, we need an algorithm which can identify any causal effect lacking a hedge. This algorithm, which we call **ID**, and which can be viewed as a simplified version of the identification algorithm due to Tian (2002), appears in Fig. 4.

We will explain why each line of **ID** makes sense, and conclude by showing the operation of the algorithm on an example. The formal proof of soundness of **ID** can be found in the appendix. The first line merely asserts that if no action has been taken, the effect on **Y** is just the marginal of the observational distribution $P(\mathbf{v})$ on **Y**. The second line states that if we are interested in the effect on **Y**, it is sufficient to restrict our attention on the parts of the model ancestral to **Y**. One intuitive argument for this is that descendants of **Y** can be viewed as 'noisy versions' of **Y** and so any information they may impart which may be helpful for identification is already present in **Y**. On the other hand, variables which are neither ancestors nor descendants of **Y** lie outside the relevant causal chain entirely, and have no useful information to contribute.

Line 3 forces an action on any node where such an action would have no effect on **Y**—assuming we already acted on **X**. Since actions remove incoming arrows, we can view line 3 as simplifying

function **ID**(**y**, **x**, P, G)
INPUT: **x**,**y** value assignments, P a probability distribution, G a causal diagram.
OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y})$ in terms of P or **FAIL**(F,F').

1 if $\mathbf{x} = \emptyset$ return $\sum_{\mathbf{v}\setminus\mathbf{y}} P(\mathbf{v})$.

2 if $\mathbf{V} \setminus An(\mathbf{Y})_G \neq \emptyset$
return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cap An(\mathbf{Y})_G, \sum_{\mathbf{V}\setminus An(\mathbf{Y})_G} P, G_{An(\mathbf{Y})})$.

3 let $\mathbf{W} = (\mathbf{V} \setminus \mathbf{X}) \setminus An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$.
if $\mathbf{W} \neq \emptyset$, return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cup \mathbf{w}, P, G)$.

4 if $C(G \setminus \mathbf{X}) = \{S_1, ..., S_k\}$
return $\sum_{\mathbf{v}\setminus(\mathbf{y}\cup\mathbf{x})} \prod_i \mathbf{ID}(s_i, \mathbf{v} \setminus s_i, P, G)$.

if $C(G \setminus \mathbf{X}) = \{S\}$

5 if $C(G) = \{G\}$, throw $\mathbf{FAIL}(G, G \cap S)$.

6 if $S \in C(G)$ return $\sum_{s\setminus\mathbf{y}} \prod_{\{i|V_i \in S\}} P(v_i | v_\pi^{(i-1)})$.

7 if $(\exists S')S \subset S' \in C(G)$ return $\mathbf{ID}(\mathbf{y}, \mathbf{x} \cap S',$
$\prod_{\{i|V_i \in S'\}} P(V_i | V_\pi^{(i-1)} \cap S', v_\pi^{(i-1)} \setminus S'), G_{S'})$.

Figure 4: A complete identification algorithm. **FAIL** propagates through recursive calls like an exception, and returns the hedge which witnesses non-identifiability. $V_\pi^{(i-1)}$ is the set of nodes preceding V_i in some topological ordering π in G.

the causal graph we consider by removing certain arcs from the graph, without affecting the overall answer. Line 4 is the key line of the algorithm, it decomposes the problem into a set of smaller problems using the key property of *c-component factorization* of causal models. If the entire graph is a single C-component already, further problem decomposition is impossible, and we must provide base cases. **ID** has three base cases. Line 5 fails because it finds two C-components, the graph G itself, and a subgraph S that does not contain any $\mathbf{X}$ nodes. But that is exactly one of the properties of C-forests that make up a hedge. In fact, it turns out that it is always possible to recover a hedge from these two c-components. Line 6 asserts that if there are no bidirected arcs from $\mathbf{X}$ to the other nodes in the current subproblem under consideration, then we can replace acting on $\mathbf{X}$ by conditioning, and thus solve the subproblem. Line 7 is the most complex case where $\mathbf{X}$ is partitioned into two sets, $\mathbf{W}$ which contain bidirected arcs into other nodes in the subproblem, and $\mathbf{Z}$ which do not. In this situation, identifying $P(\mathbf{y}|do(\mathbf{x}))$ from $P(\mathbf{v})$ is equivalent to identifying $P(\mathbf{y}|do(\mathbf{w}))$ from $P(\mathbf{V}|do(\mathbf{z}))$, since $P(\mathbf{y}|do(\mathbf{x})) = P(\mathbf{y}|do(\mathbf{w}), do(\mathbf{z}))$. But the term $P(\mathbf{V}|do(\mathbf{z}))$ is identifiable using the previous base case, so we can consider the subproblem of identifying $P(\mathbf{y}|do(\mathbf{w}))$.

We give an example of the operation of the algorithm by identifying $P_x(y_1, y_2)$ from $P(\mathbf{v})$ in the graph shown in in Fig. 3 (a). Since $G = G_{An(\{Y_1,Y_2\})}, C(G \setminus \{X\}) = \{G\}$, and $\mathbf{W} = \{W_1\}$, we invoke line 3 and attempt to identify $P_{x,w}(y_1, y_2)$. Now $C(G \setminus \{X, W\}) = \{Y_1, W_2 \to Y_2\}$, so we invoke line

Figure 5: Subgraphs of G used for identifying $P_x(y_1, y_2)$.

4. Thus the original problem reduces to identifying $\sum_{w_2} P_{x,w_1,w_2,y_2}(y_1) P_{w,x,y_1}(w_2, y_2)$. Solving for the second expression, we trigger line 2, noting that we can ignore nodes which are not ancestors of W_2 and Y_2, which means $P_{w,x,y_1}(w_2, y_2) = P(w_2, y_2)$. Solving for the first expression, we first trigger line 2 also, obtaining $P_{x,w_1,w_2,y_2}(y_1) = P_{x,w}(y_1)$. The corresponding G is shown in Fig. 5 (a). Next, we trigger line 7, reducing the problem to computing $P_w(y_1)$ from $P(Y_1|X, W_1)P(W_1)$. The corresponding G is shown in Fig. 5 (b). Finally, we trigger line 2, obtaining $P_w(y_1) = \sum_{w_1} P(y_1|x, w_1)P(w_1)$. Putting everything together, we obtain: $P_x(y_1, y_2) = \sum_{w_2} P(y_1, w_2) \sum_{w_1} P(y_1|x, w_1)P(w_1)$.

As mentioned earlier, whenever the algorithm fails at line 5, it is possible to recover a hedge from the C-components S and G considered for the subproblem where the failure occurs. In fact, it can be shown that this hedge implies the non-identifiability of the original query with which the algorithm was invoked, which implies the following result.

Theorem 18 ***ID*** *is complete.*

The completeness of **ID** implies that hedges can be used to characterize all cases where effects of the form $P(\mathbf{y}|do(\mathbf{x}))$ cannot be identified from the observational distribution $P(\mathbf{v})$.

Theorem 19 (hedge criterion) $P(\boldsymbol{v}), G \not\vdash_{id} P(\boldsymbol{y}|do(\boldsymbol{x}))$ *if and only if* G *contains a hedge for some* $P(\boldsymbol{y}'|do(\boldsymbol{x}'))$, *where* $\boldsymbol{y}' \subseteq \boldsymbol{y}$, $\boldsymbol{x}' \subseteq \boldsymbol{x}$.

We close this section by considering identification of *conditional effects* of the form $P(\mathbf{y}|do(\mathbf{x}), \mathbf{z})$ which are defined to be equal to $P(\mathbf{y}, \mathbf{z}|do(\mathbf{x}))/P(\mathbf{z}|do(\mathbf{x}))$. Such expressions are a formalization of an intuitive notion of "effect of action in the presence of non-contradictory evidence," for instance the effect of smoking on lung cancer incidence rates in a particular age group (as opposed to the effect of smoking on cancer in the general population). We say that evidence $\mathbf{z}$ is non-contradictory since it is conceivable to consider questions where the evidence $\mathbf{z}$ stands in logical contradiction to the proposed hypothetical action $do(\mathbf{x})$: for instance what is the effect of smoking on cancer among the non-smokers. Such counterfactual questions will be considered in the next section. Conditioning can both help and hinder identifiability. $P(y|do(x))$ is not identifiable in the graph shown in Fig. 6 (a), while it is identifiable in the graph shown in Fig. 6 (b). Conditioning reverses the situation. In Fig. 6 (a), conditioning on Z renders Y independent of any changes to X, making $P_x(y|z)$ equal to $P(y|z)$. On the other hand, in Fig. 6 (b), conditioning on Z makes X and Y dependent, resulting in $P_x(y|z)$ becoming non-identifiable.

We would like to reduce the problem of identifying conditional effects to the familiar problem of identifying causal effects without evidence for which we already have a complete algorithm. Fortunately, rule 2 of do-calculus provides us with a convenient way of converting the unwanted evidence $\mathbf{z}$ into actions $do(\mathbf{x})$ which we know how to handle. The following convenient lemma allows us to remove as many evidence variables as possible from a conditional effect.

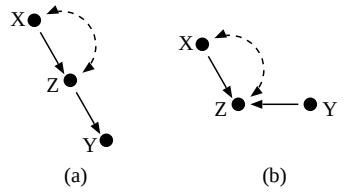


Figure 6: (a) Causal graph with an identifiable conditional effect $P(y|do(x),z)$. (b) Causal graph with a non-identifiable conditional effect $P(y|do(x),z)$.

function **IDC**(**y**, **x**, **z**, P, G)
INPUT: **x**,**y**,**z** value assignments, P a probability distribution, G a causal diagram (an I-map of P).
OUTPUT: Expression for $P_{\mathbf{x}}(\mathbf{y}|\mathbf{z})$ in terms of P or **FAIL**(F,F').

1 if $(\exists Z \in \mathbf{Z})(\mathbf{Y} \perp\!\!\!\perp Z|\mathbf{X},\mathbf{Z}\setminus\{Z\})_{G_{\overline{\mathbf{X}},\underline{z}}}$,
return $\mathbf{IDC}(\mathbf{y},\mathbf{x}\cup\{z\},\mathbf{z}\setminus\{z\},P,G)$.

2 else let $P' = \mathbf{ID}(\mathbf{y}\cup\mathbf{z},\mathbf{x},P,G)$.
return $P'/\sum_{\mathbf{y}} P'$.

Figure 7: A complete identification algorithm for conditional effects.

Theorem 20 *For any G and any conditional effect $P_{\boldsymbol{x}}(\boldsymbol{y}|\boldsymbol{w})$ there exists a unique maximal set $\boldsymbol{Z} = \{Z \in \boldsymbol{W}|P_{\boldsymbol{x}}(\boldsymbol{y}|\boldsymbol{w}) = P_{\boldsymbol{x},z}(\boldsymbol{y}|\boldsymbol{w}\setminus\{z\})\}$ such that rule 2 applies to $\boldsymbol{Z}$ in G for $P_{\boldsymbol{x}}(\boldsymbol{y}|\boldsymbol{w})$. In other words, $P_{\boldsymbol{x}}(\boldsymbol{y}|\boldsymbol{w}) = P_{\boldsymbol{x},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{w}\setminus\boldsymbol{z})$.*

Of course Theorem 20 does not guarantee that the entire set **z** can be handled in this way. In many cases, even after rule 2 is applied, some set of evidence will remain in the expression. Fortunately, the following result implies that identification of unconditional causal effects is all we need.

Theorem 21 *Let $\boldsymbol{Z} \subseteq \boldsymbol{W}$ be the maximal set such that $P_{\boldsymbol{x}}(\boldsymbol{y}|\boldsymbol{w}) = P_{\boldsymbol{x},\boldsymbol{z}}(\boldsymbol{y}|\boldsymbol{w}\setminus\boldsymbol{z})$. Then $P_{\boldsymbol{x}}(\boldsymbol{y}|\boldsymbol{w})$ is identifiable in G if and only if $P_{\boldsymbol{x},\boldsymbol{z}}(\boldsymbol{y},\boldsymbol{w}\setminus\boldsymbol{z})$ is identifiable in G.*

The previous two theorems suggest a simple addition to **ID**, which we call **IDC**, shown in Fig. 7, which handles identification of conditional causal effects.

Theorem 22 ***IDC** is sound and complete.*

Proof This follows from Theorems 20 and 21. ■

We conclude this section by showing that our notion of a causal theory as a set of independencies embodied by the causal graph, together with rules of probability and do-calculus is complete for computing causal effects, if we also take statistical data embodied by $P(\mathbf{v})$ as axiomatic.

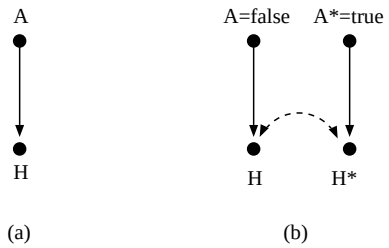


Figure 8: (a) A causal graph for the aspirin/headache domain (b) A corresponding twin network graph for the query $P(H^*_{a^*=true}|A = false)$.

Theorem 23 *The rules of do-calculus are complete for identifying effects of the form* $P(\boldsymbol{y}|do(\boldsymbol{x}),\boldsymbol{z})$, *where* $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}$ *are arbitrary sets.*

Proof The proofs of soundness of **ID** and **IDC** in the appendix use do-calculus. This implies every line of the algorithms we presented can be rephrased as a sequence of do-calculus manipulations. But **ID** and **IDC** are also complete, which implies the conclusion. ■

4. Identification of Counterfactuals

While effects of actions have an intuitive interpretation as downward flow, the interpretation of counterfactuals, or what-if questions is more complex. An informal counterfactual statement in natural language such as "would I have a headache had I taken an aspirin" talks about multiple worlds: the actual world, and other, hypothetical worlds which differ in some small respect from the actual world (e.g., the aspirin was taken), while in most other respects are the same. In this paper, we represent the actual world by a causal model in its natural state, devoid of any interventions, while the alternative worlds are represented by submodels $M_{\mathbf{x}}$ where the action $do(\mathbf{x})$ implements the hypothetical change from the actual state of affairs considered. People make sense of informal statements involving multiple, possibly conflicting worlds because they expect not only the causal rules to be invariant across these worlds (e.g., aspirin helps headaches in all worlds), but the worlds themselves to be similar enough where evidence in one world has ramifications in another. For instance, if we find ourselves with a headache, we expect the usual causes of our headache to also operate in the hypothetical world, interacting there with the preventative influence of aspirin. In our representation of counterfactuals, we model this interaction between worlds by assuming that the world histories or background contexts, represented by the unobserved **U** variables are shared across all hypothetical worlds.

We illustrate the representation method for counterfactuals we introduced in Section 2 by modeling our example question "would I have a headache had I taken an aspirin?" The actual world referenced by this query is represented by a causal model containing two variables, headache and aspirin, with aspirin being a parent of headache, see Fig. 8 (a). In this world, we observe that aspirin has value false. The hypothetical world is represented by a submodel where the action $do(aspirin = true)$ has been taken. To distinguish nodes in this world we augment their names with an asterisk. The two worlds share the background variables **U**, and so can be represented by a

single causal model with the graph shown in Fig. 8 (b). Our query is represented by the distribution $P(H^*_{a^*=true}|A = false)$, where H is headache, and A is aspirin. Note that the nodes $A^* = true$ and $A = false$ in Fig. 8 (b) do not share a bidirected arc. This is because an intervention $do(a^* = true)$ removes all incoming arrows to A^*, which removes the bidirected arc between A^* and A.

The graphs representing two hypothetical worlds invoked by a counterfactual query like the one shown in Fig. 8 (b) are called *twin network graphs*, and were first proposed as a way to represent counterfactuals by Balke and Pearl (1994b) and Balke and Pearl (1994a). In addition, Balke and Pearl (1994b) proposed a method for evaluating expressions like $P(H^*_{a^*=true}|A = false)$ when all parameters of a causal model are known. This method can be explained as follows. If we forget the causal and counterfactual meaning behind the twin network graph, and simply view it as a Bayesian network, the query $P(H^*_{a^*=true}|A = false)$ can be evaluated using any of the standard inference algorithms available, provided we have access to all conditional probability tables generated by $\mathbf{F}$ and $\mathbf{U}$ of a causal model which gave rise to the twin network graph. In practice, however, complete knowledge of the model is too much to ask for; the functional relationships as well as the distribution $P(\mathbf{u})$ are not known exactly, though some of their aspects can be inferred from the observable distribution $P(\mathbf{v})$.

Instead, the typical state of knowledge of a causal domain is the statistical behavior of the observable variables in the domain, summarized by the distribution $P(\mathbf{v})$, together with knowledge of causal directionality, obtained either from expert judgment (e.g., we know that visiting the doctor does not make us sick, though disease and doctor visits are highly correlated), or direct experimentation (e.g., it's easy to imagine an experiment which establishes that wet grass does not cause sprinklers to turn on). We already used these two sources of knowledge in the previous section as a basis for computing causal effects. Nevertheless, there are reasons to consider computing counterfactual quantities from experimental, rather than observational studies. In general, a counterfactual can posit worlds with features contradictory to what has actually been observed. For instance, questions resembling the headache/aspirin question we used as an example are actually frequently asked in epidemiology in the more general form where we are interested in estimating the effect of a treatment x on the outcome variable Y for the patients that were not treated (x'). In our notation, this is just our familiar expression $P(Y_x|X = x')$. The problem with questions such as these is that no experimental setup exists in which someone is both given and not given treatment. Therefore, it makes sense to ask under what circumstances we can evaluate such questions even if we are given as input every experiment that is possible to perform in principle on a given causal model. In our framework the set of all experiments is denoted as P_*, and is formally defined as $\{P_{\mathbf{x}}|\mathbf{x}$ is any set of values of $\mathbf{X} \subseteq \mathbf{V}\}$. The question that we ask in this section, then, is whether it is possible to identify a query $P(\gamma|\delta)$, where γ, δ are conjunctions of counterfactual events (with δ possibly empty), from the graph G and the set of all experiments P_*. We can pose the problem in this way without loss of generality since we already developed complete methods for identifying members of P_* from G and $P(\mathbf{v})$. This means that if for some reason using P_* as input is not realistic we can combine the methods which we will develop in this section with those in the previous section to obtain identification results for $P(\gamma|\delta)$ from G and $P(\mathbf{v})$.

Before tackling the problem of identifying counterfactual queries from experiments, we extend our example in Fig. 8 (b) to a general graphical representation for worlds invoked by a counterfactual query. The twin network graph is a good first attempt at such a representation. It is essentially a causal diagram for a model encompassing two potential worlds. Nevertheless, the twin network graph suffers from a number of problems. Firstly, it can easily come to pass that a counterfactual

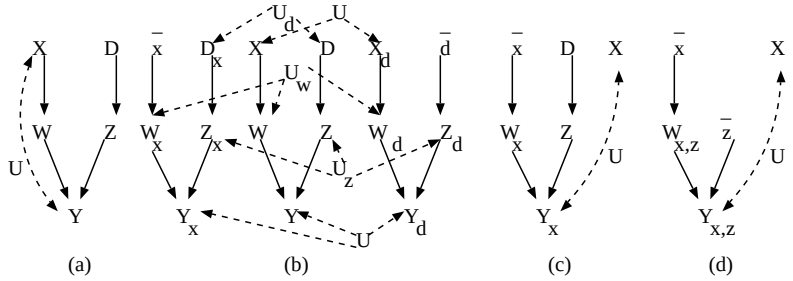


Figure 9: Nodes fixed by actions denoted with an overline, signifying that all incoming arrows are cut. (a) Original causal diagram (b) Parallel worlds graph for $P(y_x|x',z_d,d)$ (the two nodes denoted by U are the same). (c) Counterfactual graph for $P(y_x|x',z_d,d)$. (d) Counterfactual graph for $P(y_{x,z}|x')$.

query of interest would involve three or more worlds. For instance, we might be interested in how likely the patient would be to have a symptom Y given a certain dose x of drug X, assuming we know that the patient has taken dose x' of drug X, dose d of drug D, and we know how an intermediate symptom Z responds to treatment d. This would correspond to the query $P(y_x|x',z_d,d)$, which mentions three worlds, the original model M, and the submodels M_d, M_x. This problem is easy to tackle—we simply add more than two submodel graphs, and have them all share the same **U** nodes. This simple generalization of the twin network model was considered by Avin et al. (2005), and was called there the parallel worlds graph. Fig. 9 shows the original causal graph and the parallel worlds graph for $\gamma = y_x \wedge x' \wedge z_d \wedge d$.

The other problematic feature of the twin network graph, which is inherited by the parallel worlds graph, is that multiple nodes can sometimes correspond to the same random variable. For example, in Fig. 9 (b), the variables Z and Z_x are represented by distinct nodes, although it's easy to show that since Z is not a descendant of X, $Z = Z_x$. These equality constraints among nodes can make the d-separation criterion misleading if not used carefully. For instance, $Y_x \perp\!\!\!\perp D_x|Z$ even though using d-separation in the parallel worlds graph suggests the opposite. This sort of problem is fairly common in causal models which are not *faithful* (Spirtes et al., 1993) or *stable* (Pearl, 2000), in other words in models where d-separation statements in a causal diagram imply independence in a distribution, but not vice versa. However, lack of faithfulness usually arises due to "numeric coincidences" in the observable distribution. In this case, the lack of faithfulness is "structural," in a sense that it is possible to refine parallel worlds graphs in such a way that the node duplication disappears, and the attendant independencies not captured by d-separation are captured by d-separation in refined graphs.

This refinement has two additional beneficial side effects. The first is that by removing node duplication, we also determine which syntactically distinct counterfactual variables correspond to the same random variable. By identifying such equivalence classes of counterfactual variables, we guarantee that syntactically different variables are in fact different, and this makes it simpler to reason about counterfactuals in order to identify them. For instance, a counterfactual $P(y_x,y')$ may either be non-identifiable or inconsistent (and so identifiable to equal 0), depending on whether Y_x and Y are the same variable. The second benefit of this refinement is that resulting graphs are gen-

erally much smaller and less cluttered than parallel worlds graphs, and so are easier to understand. Compare, for instance, the graphs in Fig. 9 (b) and Fig. 9 (c). To rid ourselves of duplicates, we need a formal way of determining when variables from different submodels are in fact the same. The following lemma does this.

Lemma 24 *Let M be a model inducing G containing variables α, β with the following properties:*

- *α and β have the same domain of values.*
- *There is a bijection f from $Pa(\alpha)$ to $Pa(\beta)$ such that a parent γ and $f(\gamma)$ have the same domain of values.*
- *The functional mechanisms of α and β are the same (except whenever the function for α uses the parent γ, the corresponding function for β uses $f(\gamma)$).*

Assume an observable variable set $\mathbf{Z}$ was observed to attain values $\mathbf{z}$ in $M_{\mathbf{X}}$, the submodel obtained from M by forcing another observable variable set $\mathbf{X}$ to attain values $\mathbf{x}$. Assume further that for each $\gamma \in Pa(\alpha)$, either $f(\gamma) = \gamma$, or γ and $f(\gamma)$ attain the same values (whether by observation or intervention). Then α and β are the same random variable in $M_{\mathbf{X}}$ with observations $\mathbf{z}$.

Proof This follows from the fact that variables in a causal model are functionally determined from their parents. ■

If two distinct nodes in a causal diagram represent the same random variable, the diagram contains redundant information, and the nodes must be merged. If two nodes, say corresponding to $Y_{\mathbf{x}}, Y_{\mathbf{z}}$, are established to be the same in G, they are merged into a single node which inherits all the children of the original two. These two nodes either share their parents (by induction) or their parents attain the same values. If a given parent is shared, it becomes the parent of the new node. Otherwise, we pick one of the parents arbitrarily to become the parent of the new node. This operation is summarized by the following lemma.

Lemma 25 *Let $M_{\mathbf{X}}$ be a submodel derived from M with set $\mathbf{Z}$ observed to attain values $\mathbf{z}$, such that Lemma 24 holds for α, β. Let M' be a causal model obtained from M by merging α, β into a new node ω, which inherits all parents and the functional mechanism of α. All children of α, β in M' become children of ω. Then $M_{\mathbf{X}}, M'_{\mathbf{X}}$ agree on any distribution consistent with $\mathbf{z}$ being observed.*

Proof This is a direct consequence of Lemma 24. ■

The new node ω we obtain from Lemma 25 can be thought of as a new counterfactual variable. As mentioned in section 2, such variables take the form $Y_{\mathbf{x}}$ where Y is the variable in the original causal model, and $\mathbf{x}$ is a subscript specifying the action which distinguishes the counterfactual. Since we only merge two variables derived from the same original, specifying Y is simple. But what about the subscript? Intuitively, the subscript of ω contains those fixed variables which are ancestors of ω in the graph G' of M'. Formally the subscript is $\mathbf{w}$, where $\mathbf{W} = An(\omega)_{G'} \cap \mathbf{sub}(\gamma)$, where the $\mathbf{sub}(\gamma)$ corresponds to those nodes in G' which correspond to subscripts in γ. Since we replaced α, β by ω, we replace any mention of α, β in our given counterfactual query $P(\gamma)$ by ω.

function **make-cg**(G, γ)
INPUT: G a causal diagram, γ a conjunction of counterfactual events
OUTPUT: A counterfactual graph G_γ, and either a set of events γ' s.t. $P(\gamma') = P(\gamma)$ or **INCONSISTENT**

- Construct a submodel graph $G_{\mathbf{x}_i}$ for each action $do(\mathbf{x}_i)$ mentioned in γ. Construct the parallel worlds graph G' by having all such submodel graphs share their corresponding U nodes.
- Let π be a topological ordering of nodes in G', let $\gamma' := \gamma$.
- Apply Lemmas 24 and 25, in order π, to each observable node pair α, β derived from the same variable in G. For each α, β that are the same, do:
  - Let G' be modified as specified in Lemma 25.
  - Modify γ' by renaming all occurrences of β to α.
  - If $\mathbf{val}(\alpha) \neq \mathbf{val}(\beta)$, return $G', \mathbf{INCONSISTENT}$.
- return $(G'_{An(\gamma')}, \gamma')$, where $An(\gamma')$ is the set of nodes in G' ancestral to nodes corresponding to variables mentioned in γ'.

Figure 10: An algorithm for constructing counterfactual graphs

Note that since α, β are the *same*, their value assignments must be the same (say equal to y). The new counterfactual ω inherits this assignment.

We summarize the inductive applications of Lemma 24, and 25 by the **make-cg** algorithm, which takes γ and G as arguments, and constructs a version of the parallel worlds graph without duplicate nodes. We call the resulting structure the *counterfactual graph* of γ, and denote it by G_γ. The algorithm is shown in Fig. 10.

There are three additional subtleties in **make-cg**. The first is that if variables $Y_{\mathbf{X}}, Y_{\mathbf{Z}}$ were judged to be the same by Lemma 24, but γ assigns them different values, this implies that the original set of counterfactual events γ is inconsistent, and so $P(\gamma) = 0$. The second is that if we are interested in identifiability of $P(\gamma)$, we can restrict ourselves to the ancestors of γ in G'. We can justify this using the same intuitive argument we used in Section 3 to justify Line 2 in **ID**. The formal proof for line 2 we provide in the appendix applies with little change to **make-cg**. Finally, because the algorithm can make an arbitrary choice picking a parent of ω each time Lemma 25 is applied, both the counterfactual graph G', and the corresponding modified counterfactual γ' are not unique. This does not present a problem, however, as any such graph is acceptable for our purposes.

We illustrate the operation of **make-cg** by showing how the graph in Fig. 9 (c) is derived from the graph in Fig. 9 (b). We start the application of Lemma 24 from the topmost observable nodes, and conclude that the node pairs D_x, D, and X_d, X have the same functional mechanisms, and the same parent set (in this case the parents are unobservable nodes U_d for the first pair, and U for the second). We then use Lemma 25 to obtain the graph shown in Fig. 11 (a). Since the node pairs are the same, we pick the name of one of the nodes of the pair to serve as the name of the new node. In our case, we picked D and X. Note that for this graph, and all subsequent intermediate graphs we generate, we use the convention that if a merge creates a situation where an unobservable variable

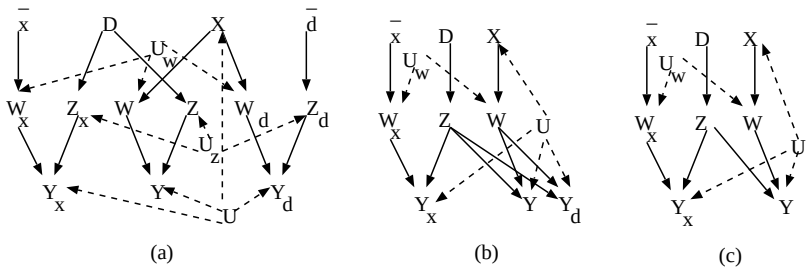


Figure 11: Intermediate graphs obtained by **make-cg** in constructing the counterfactual graph for $P(y_x|x',z_d,d)$ from Fig. 9 (b).

has a single child, that variable is omitted from the graph. For instance, in Fig. 11 (a), the variable U_d, and its corresponding arrow to D omitted.

Next, we apply Lemma 24 for the node pair W_d, W. In this case, the functional mechanisms are once again the same, while the parents of W_d, W are X and U_w. We can also apply Lemma 24 twice to conclude that Z, Z_x and Z_d are in fact the same node, and so can be merged. The functional mechanisms of these three nodes are the same, and they share the parent U_z. As far as the parents of this triplet, the U_z parent is shared by all three, while Z, Z_x share the parent D, and Z_d has a separate parent d, fixed by intervention. However, in our counterfactual query, which is $P(y_x|x',z_d,d)$, the variable D happens to be observed to attain the value d, the same as the intervention value for the parent of Z_d. This implies that for the purposes of the Z, Z_x, Z_d triplet, their D-derived parents share the same value, which allows us to conclude they are the same random variable. The intuition here is that while intervention and observation are not the same operation, they have the same effect if the relevant U variables happen to react in the same way to both the given intervention, and the given observation (this is the essence of the Axiom of Composition discussed by Pearl (2000).) In our case, U variables react the same way because the parallel worlds share all unobserved variables.

There is one additional subtlety in performing the merge of the triplet Z, Z_x, Z_d. If we examine our query $P(y_x|x',z_d,d)$, we notice that Z_d, or more precisely its value, appears in it. When we merge nodes, we only use one name out of the original two. It's possible that some of the old names appear in the query, which means we must replace all references to the old, pre-merge nodes with the new post-merge name we picked. Since we picked the name Z for the newly merged node, we replace the reference to Z_d in our query by the reference to Z, so our modified query is $P(y_x|x',z,d)$. Since the variables were established to be the same, this is a safe syntactic transformation.

After W_d, W, and the Z, Z_x, Z_d triplet are merged, we obtain the graph in Fig. 11 (b). Finally, we apply Lemma 24 one more time to conclude Y and Y_d are the same variable, using the same reasoning as before. After performing this final merge, we obtain the graph in Fig. 11 (c). It's easy to see that Lemma 24 no longer applies to any node pair: W and W_x differ in their X-derived parent, and Y, and Y_x differ on their W-derived parent, which was established inductively. The final operation which **make-cg** performs is restricting the graph in Fig. 11 (b) to variables actually relevant for computing the (potentially syntactically modified) query it was given as input, namely $P(y_x|x',z,d)$. These relevant variables are ancestral to variables in the query in the final intermediate graph we

function **ID***(G,γ)
INPUT: G a causal diagram, γ a conjunction of counterfactual events
OUTPUT: an expression for $P(\gamma)$ in terms of P_* or **FAIL**

1 if $\gamma = \emptyset$, return 1

2 if $(\exists x_{x'..} \in \gamma)$, return 0

3 if $(\exists x_{x..} \in \gamma)$, return **ID***$(G, \gamma \setminus \{x_{x..}\})$

4 $(G', \gamma') =$ **make-cg**(G, γ)

5 if $\gamma' =$ **INCONSISTENT**, return 0

6 if $C(G') = \{S^1, ..., S^k\}$,
return $\sum_{V(G')\setminus\gamma'} \prod_i$ **ID***$(G, s^i_{v(G')\setminus s^i})$

7 if $C(G') = \{S\}$ then,

8 if $(\exists \mathbf{x}, \mathbf{x}')$ s.t. $\mathbf{x} \neq \mathbf{x}', \mathbf{x} \in$ **sub**$(S), \mathbf{x}' \in$ **ev**(S),
throw **FAIL**

9 else, let $\mathbf{x} = \bigcup$**sub**(S)
return $P_{\mathbf{x}}($**var**$(S))$

function **IDC***(G, γ, δ)
INPUT: G a causal diagram, γ, δ conjunctions of counterfactual events
OUTPUT: an expression for $P(\gamma|\delta)$ in terms of P_*, **FAIL**, or **UNDEFINED**

1 if **ID***$(G, \delta) = 0$, return **UNDEFINED**

2 $(G', \gamma' \wedge \delta') =$ **make-cg**$(G, \gamma \wedge \delta)$

3 if $\gamma' \wedge \delta' =$ **INCONSISTENT**, return 0

4 if $(\exists y_{\mathbf{x}} \in \delta')$ s.t. $(Y_{\mathbf{x}} \perp\!\!\!\perp \gamma')_{G'_{\underline{\mathbf{x}}}}$,
return **IDC***$(G, \gamma'_{y_{\mathbf{x}}}, \delta' \setminus \{y_{\mathbf{x}}\})$

5 else, let $P' =$ **ID***$(G, \gamma' \wedge \delta')$. return $P'/P'(\delta)$

Figure 12: Counterfactual identification algorithms.

obtained. In our case, we remove nodes W and Y (and their adjacent edges) from consideration, to finally obtain the graph in Fig. 9 (c), which is a counterfactual graph for our query.

Having constructed a graphical representation of worlds mentioned in counterfactual queries, we can turn to identification. We construct two algorithms for this task, the first is called **ID*** and works for unconditional queries, while the second, **IDC***, works on queries with counterfactual evidence and calls the first as a subroutine. These are shown in Fig. 12.

These algorithms make use of the following notation: **sub**$(.)$ returns the set of subscripts, **var**$(.)$ the set of variables, and **ev**$(.)$ the set of values (either set or observed) appearing in a given counterfactual conjunction (or set of counterfactual events), while **val**$(.)$ is the value assigned to a given counterfactual variable. This notation is used to extract variables and values present in the original causal model from a counterfactual which refers to parallel worlds. As before, $C(G')$ is the set of maximal C-components of G', except we don't count nodes in G' fixed by interventions as part of any C-component. $V(G')$ is the set of observable nodes of G' not fixed by interventions. Following Pearl (2000), $G'_{\underline{y_{\mathbf{X}}}}$ is the graph obtained from G' by removing all outgoing arcs from $Y_{\mathbf{X}}$; $\gamma'_{y_{\mathbf{X}}}$ is obtained from γ' by replacing all descendant variables $W_{\mathbf{Z}}$ of $Y_{\mathbf{X}}$ in γ' by $W_{\mathbf{Z},y}$. A counterfactual $\mathbf{s_r}$, where $\mathbf{s}, \mathbf{r}$ are value assignments to sets of nodes, represents the event "the node set $\mathbf{S}$ attains values $\mathbf{s}$ under intervention $do(\mathbf{r})$." For instance, the term $s^i_{v(g')\setminus s^i}$ stands for the event "the node set S^i attains values s^i under the intervention $do(v(g')\setminus s^i)$," in other words under the intervention where we fix the values of all observable nodes in G' except those in S^i. Finally, we take $x_{x..}$ to mean some counterfactual variable derived from X where x appears in the subscript (the rest of the subscript can be arbitrary), which also attains value x.

The notation used in these algorithms is somewhat intricate, so we give an intuitive description of each line. We start with **ID***. The first line states that if γ is an empty conjunction, then its probability is 1, by convention. The second line states that if γ contains a counterfactual which violates the Axiom of Effectiveness (Pearl, 2000), then γ is inconsistent, and we return probability 0. The third line states that if a counterfactual contains its own value in the subscript, then it is a tautological event, and it can be removed from γ without affecting its probability. Line 4 invokes **make-cg** to construct a counterfactual graph G', and the corresponding relabeled counterfactual γ'. Line 5 returns probability 0 if an inconsistency was found during the construction of the counterfactual graph, for example, if two variables found to be the same in γ had different value assignments. Line 6 is analogous to Line 4 in the **ID** algorithm, it decomposes the problem into a set of subproblems, one for each C-component in the counterfactual graph. In the **ID** algorithm, the term corresponding to a given C-component S_i of the causal diagram was the effect of all variables not in S_i on variables in S_i, in other words $P_{\mathbf{V}\setminus s_i}(s_i)$, and the outermost summation on line 4 was over values of variables not in $\mathbf{Y}, \mathbf{X}$. Here, the term corresponding to a given C-component S^i of the counterfactual graph G' is the conjunction of counterfactual variables where each variable contains in its subscript all variables not in the C-component S^i, in other words $\mathbf{v}(G')\setminus s^i$, and the outermost summation is over observable variables not in γ', that is over $\mathbf{v}(G')\setminus\gamma'$, where we interpret γ' as a set of counterfactuals, rather than a conjunction. Line 7 is the base case, where our counterfactual graph has a single C-component. There are two cases, corresponding to line 8 and line 9. Line 8 says that if γ' contains a "conflict," that is an inconsistent value assignment where at least one value is in the subscript, then we fail. Line 9 says if there are no conflicts, then its safe to take the union of all subscripts in γ', and return the effect of the subscripts in γ' on the variables in γ'.

The **IDC***, like its counterpart **IDC**, is shorter. The first line fails if δ is inconsistent. **IDC** did not have an equivalent line, since we can assume $P(\mathbf{v})$ is positive. The problem with counterfactual distributions is there is no simple way to prevent non-positive distributions spanning multiple worlds from arising, even if the original $P(\mathbf{v})$ was positive—hence the explicit check. The second line constructs the counterfactual graph, except since **make-cg** can only take conjunctions, we provide it with a joint counterfactual $\gamma\wedge\delta$. Line 3 returns 0 if an inconsistency was detected. Line 4 of **IDC*** is the central line of the algorithm and is analogous to line 1 of **IDC**. In **IDC**, we moved a

value assignment $Z = z$ from being observed to being fixed if there were no back-door paths from Z to the outcome variables $\mathbf{Y}$ given the context of the effect of $do(\mathbf{x})$. Here in **IDC***, we move a counterfactual value assignment $Y_{\mathbf{X}} = y$ from being observed (that is being a part of δ), to being fixed (that is appearing in every subscript of γ') if there are no back-door paths from $Y_{\mathbf{X}}$ to the counterfactual of interest γ'. Finally, line 5 of **IDC*** is the analogue of line 2 of **IDC**, we attempt to identify a joint counterfactual probability, and then obtain a conditional counterfactual probability from the result.

We illustrate the operation of these algorithms by considering the identification of a query $P(y_x|x',z_d,d)$ we mentioned earlier. Since $P(x',z_d,d)$ is not inconsistent, we proceed to construct the counterfactual graph on line 2. Suppose we produce the graph in Fig. 9 (c), where the corresponding modified query is $P(y_x|x',z,d)$. Since $P(y_x,x',z,d)$ is not inconsistent we proceed to the next line, which moves z,d (with d being redundant due to graph structure) to the subscript of y_x, to obtain $P(y_{x,z}|x')$, and calls **IDC*** with this query recursively. Note that since the subscripts in one of the variables of our query changed, the counterfactual graph generated will change as well. In particular, the invocation of **make-cg** with the joint distribution from which $P(y_{x,z}|x')$ is derived, namely $P(y_{x,z},x')$, will result in the graph shown in Fig. 9 (d). Since X' has a back-door path to $Y_{x,z}$ in this graph, we can no longer call **IDC*** recursively, so we invoke **ID*** with the query $P(y_{x,z},x')$.

The first interesting line in **ID*** is line 6, which first computes $P(y_{x,z},w_{x,z},x')$ by C-component factorization, and then computes $P(y_{x,z},x')$ from $P(y_{x,z},w_{x,z},x')$ by marginalizing over $W_{x,z}$.[2] Since the counterfactual graph for this query (Fig. 9 (d)) has two C-components, $\{Y_{x,z},X\}$ and $\{W_{x,z}\}$, $P(y_{x,z},w_{x,z},x') = P(y_{x,z,w},x'_w)P(w_{x,z})$, which can be simplified by removing redundant subscripts to $P(y_{z,w},x')P(w_x)$. Line 6 then recursively calls **ID*** with $P(y_{x,z,w},x')$ and $P(w_x)$, multiplies the results and marginalizes over W_x. The first recursive call reaches line 9 with $P(y_{z,w},x')$, which is identifiable as $P_{z,w}(y,x')$ from P_*. The second term is trivially identifiable as $P_x(w)$, which means our query is identifiable as $P' = \sum_w P_{z,w}(y,x')P_x(w)$, and the conditional query is equal to $P'/P'(x')$.

The definitions of **ID***, and **IDC*** reveal their close similarity to algorithms **ID** and **IDC** in the previous section. The major differences lie in the failure and success base cases, and slightly different subscript notation. This is not a coincidence, since a counterfactual graph can be thought of as a causal graph for a particular large causal model which happens to have some distinct nodes share the same causal mechanisms. This means that all the theorems and definitions used in the previous sections for causal diagrams transfer over without change to counterfactual graphs. Using this fact, we will show that **ID***, and **IDC*** are sound and complete for identifying $P(\gamma)$, and $P(\gamma|\delta)$ respectively.

Theorem 26 (soundness) *If **ID*** succeeds, the expression it returns is equal to $P(\gamma)$ in a given causal graph. Furthermore, if **IDC*** does not output **FAIL**, the expression it returns is equal to $P(\gamma|\delta)$ in a given causal graph, if that expression is defined, and **UNDEFINED** otherwise.*

Proof outline The first line merely states that the probability of an empty conjunction is 1, which is true by convention. Lines 2 and 3 follow by the Axiom of Effectiveness (Galles and Pearl, 1998). The soundness of **make-cg** has already been established, which implies the soundness of line 4. Line 6 decomposes the problem using c-component factorization. The soundness proof for this decomposition, also used in the previous section, is in the appendix. Line 9 asserts that if a set

2. Note that since $W_{x,z}$ is a counterfactual variable derived from W, it shares its domain with W. Therefore it makes sense when marginalizing to operate over the values of W, denoted by w in the subscript of the summation.

of counterfactual events does not contain conflicting value assignments to any variable, obtained either by observation or intervention, then taking the union of all actions of the events results in a consistent action. The probability of the set of events can then be computed from a submodel where this consistent action has taken place. A full proof of this is in the appendix. ■

To show completeness, we follow the same strategy we used in the previous section. We catalogue all difficult counterfactual graphs which arise from queries which cannot be identified from P_*. We then show these graphs arise whenever **ID*** and **IDC*** fail. This, together with the soundness theorem we already proved, implies that these algorithms are complete.

The simplest difficult counterfactual graph arises from the query $P(y_x, y'_{x'})$ named "probability of necessity and sufficiency" by Pearl (2000). This graph, shown in Fig. 8 (b) with variable relabeling, is called the "w-graph" due to its shape (Avin et al., 2005). This query is so named because if $P(y_x, y'_{x'})$ is high, this implies that if the variable X is forced to x, variable Y is likely to be y, while if X is forced to some other value, Y is likely to not be y. This means that the action $do(x)$ is likely a necessary and sufficient cause of Y assuming value y, up to noise. The w-graph starts our catalogue of bad graphs with good reason, as the following lemma shows.

Lemma 27 *Assume X is a parent of Y in G. Then $P_*, G \not\vdash_{id} P(y_x, y'_{x'}), P(y_x, y')$ for any value pair y, y'.*

Proof See Avin et al. (2005). ■

The intuitive explanation for this result is that $P(y_x, y'_{x'})$ is derived from the joint distribution over the counterfactual variables in the w-graph, while if we restrict ourselves to P_*, we only have access to marginal distributions—one marginal for each possible world. Because counterfactual variables Y_x and $Y_{x'}$ share an unobserved parent U, they are dependent, and their joint distribution cannot be decomposed into a product of marginals. This means that the information encoded in the marginals is insufficient to uniquely determine the joint we are interested in. This intuitive argument can be generalized to a counterfactual graph with more than two nodes, the so-called "zig-zag graphs" an example of which is shown in Fig. 13 (b).

Lemma 28 *Assume G is such that X is a parent of Y and Z, and Y and Z are connected by a bidirected path with observable nodes $W^1, \ldots, W^k$ on the path. Then $P_*, G \not\vdash_{id} P(y_x, w^1, \ldots, w^k, z_{x'})$, $P(y_x, w^1, \ldots, w^k, z)$ for any value assignments $y, w^1, \ldots, w^k, z$.*

The w-graph in Fig. 8 (b) and the zig-zag graph in Fig. 13 (b) have very special structure, so we don't expect our characterization to be complete with just these graphs. In order to continue, we must provide two lemmas which allow us to transform difficult graphs in various ways by adding nodes and edges, while retaining the non-identifiability of the underlying counterfactual from P_*.

Lemma 29 (downward extension lemma) *Assume $P_*, G \not\vdash_{id} P(\gamma)$. Let $\{y^1_{\mathbf{x}^1}, \ldots, y^n_{\mathbf{x}^m}\}$ be a subset of counterfactual events in γ. Let G' be a graph obtained from G by adding a new child W of $Y^1, \ldots, Y^n$, and let P'_* be the set of all interventional distributions in models inducing G'. Let $\gamma' = (\gamma \setminus \{y^1_{\mathbf{x}^1}, \ldots, y^n_{\mathbf{x}^m}\}) \cup \{w_{\mathbf{x}^1}, \ldots, w_{\mathbf{x}^m}\}$, where w is an arbitrary value of W. Then $P'_*, G' \not\vdash_{id} P(\gamma')$.*

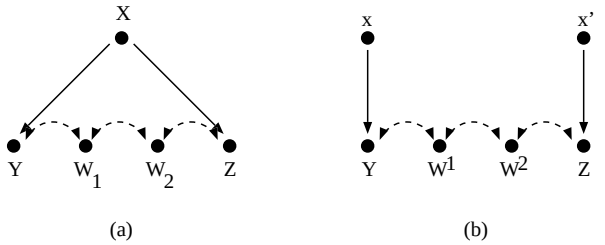


Figure 13: (a) Causal diagram (b) Corresponding counterfactual graph for the non-identifiable query $P(Y_x, W^1, W^2, Z_{x'})$.

The first result states that non-identification on a set of parents (causes) translates into non-identification on children (effects). The intuitive explanation for this is that it is possible to construct a one-to-one function from the space of distributions on causes to the space of distributions on effects. If a given $P(\gamma)$ cannot be identified from P_*, this implies that there exist two models which agree on P_*, but disagree on $P(\gamma)$, where γ is a set of counterfactual causes. It is then possible to augment these models using the one-to-one function in question to obtain disagreement on $P(\delta)$, where δ is a set of counterfactual effects of γ. A more detailed argument is found in the appendix.

Lemma 30 (contraction lemma) *Assume $P_*, G \not\vdash_{id} P(\gamma)$. Let G' be obtained from G by merging some two nodes X,Y into a new node Z where Z inherits all the parents and children of X,Y, subject to the following restrictions:*

- *The merge does not create cycles.*
- *If $(\exists w_{\mathbf{S}} \in \gamma)$ where $x \in \mathbf{s}$, $y \notin \mathbf{s}$, and $X \in An(W)_G$, then $Y \notin An(W)_G$.*
- *If $(\exists y_{\mathbf{S}} \in \gamma)$ where $x \in \mathbf{s}$, then $An(X)_G = \emptyset$.*
- *If $(Y_{\mathbf{w}}, X_{\mathbf{S}} \in \gamma)$, then $\mathbf{w}$ and $\mathbf{s}$ agree on all variable settings.*

Assume $|X| \times |Y| = |Z|$ and there's some isomorphism f assigning value pairs x,y to a value $f(x,y) = z$. Let γ' be obtained from γ as follows. For any $w_{\mathbf{S}} \in \gamma$:

- *If $W \notin \{X,Y\}$, and values x,y occur in $\mathbf{s}$, replace them by $f(x,y)$.*
- *If $W \notin \{X,Y\}$, and the value of one of X,Y occur in $\mathbf{s}$, replace it by some z consistent with the value of X or Y.*
- *If X,Y do not occur in γ, leave γ as is.*
- *If $W = Y$ and $x \in \mathbf{s}$, replace $w_{\mathbf{S}}$ by $f(x,y)_{\mathbf{S}\setminus\{x\}}$.*
- *otherwise, replace every variable pair of the form $Y_{\mathbf{r}} = y, X_{\mathbf{S}} = x$ by $Z_{\mathbf{r},\mathbf{S}} = f(x,y)$.*

Then $P_*, G' \not\vdash_{id} P(\gamma')$.

This lemma has a rather complicated statement, but the basic idea is very simple. If we have a causal model with a graph G where some counterfactual $P(\gamma)$ is not identifiable, then a coarser, more "near-sighted" view of G which merges two distinct variables with their own mechanisms into a single variable with a single mechanism will not render $P(\gamma)$ identifiable. This is because merging nodes in the graph does not alter the model, but only our state of knowledge of the model. Therefore, whatever model pair was used to prove $P(\gamma)$ non-identifiable will remain the same in the new, coarser graph. The complicated statement of the lemma is due to the fact that we cannot allow arbitrary node merges, we must satisfy certain coherence conditions. For instance, the merge cannot create directed cycles in the graph.

It turns out that whenever **ID*** fails on $P(\gamma)$, the corresponding counterfactual graph contains a subgraph which can be obtained by a set of applications of the previous two lemmas to the w-graph and the zig-zag graphs. This allows an argument that shows $P(\gamma)$ cannot be identified from P_*.

Theorem 31 (completeness) *If **ID*** or **IDC*** fail, then the corresponding query is not identifiable from P_*.*

Since **ID*** is complete for $P(\gamma)$ queries, we can give a graphical characterization of counterfactual graphs where $P(\gamma)$ cannot be identified from P_*.

Theorem 32 *Let G_γ, γ' be obtained from **make-cg**(G,γ). Then $P_*, G \not\vdash_{id} P(\gamma)$ if and only if there exists a C-component $S \subseteq An(\gamma')_{G_\gamma}$ where some $X \in Pa(S)$ is set to x while at the same time either X is also a parent of another node in S and is set to another value x', or S contains a variable derived from X which is observed to be x'.*

Proof This follows from Theorem 31 and the construction of **ID***. ■

5. Conclusions

This paper considers a hierarchy of queries about relationships among variables in graphical causal models: associational relationships which can be obtained from observational studies, cause-effect relationships obtained by experimental studies, and counterfactuals, which are derived from parallel worlds resulting from hypothetical actions, possibly conflicting with available evidence. We consider the identification problem for this hierarchy, the task of computing a query from the given causal diagram and available information lower in the hierarchy.

We provide sound and complete algorithms for this identification problem, and a graphical characterization of non-identifiable queries where these algorithms must fail. Specifically, we provide complete algorithms for identifying causal effects and conditional causal effects from observational studies, and show that a graphical structure called a *hedge* completely characterizes all cases where causal effects are non-identifiable. As a corollary, we show that the three rules of do-calculus are complete for identifying effects. We also provide complete algorithms for identifying counterfactual queries (possibly conditional) from experimental studies. If we view the structure of the causal graph as experimentally testable, as is often the case in practice, this result can be viewed as giving a full characterization of testable counterfactuals assuming structural semantics.

These results settle important questions in causal inference, and pave the way for computing more intricate causal queries which involve nested counterfactuals, such as those defining direct

and indirect effects (Pearl, 2001), and path-specific effects (Avin et al., 2005). The characterization of non-identifiable queries we provide defines precisely the situations when such queries cannot be computed precisely, and must instead by approximated using methods such as bounding (Balke and Pearl, 1994a), instrumental variables (Pearl, 2000), or additional assumptions, such as linearity, which can make identification simpler.

Acknowledgments

The authors would like to thank Eleazar Eskin and Eun Yong Kang for discussing earlier versions of this work. This work was supported in part by AFOSR grant #F49620-01-1-0055, NSF grant #IIS-0535223, MURI grant #N00014-00-1-0617, and NLM grant #T15 LM07356.

Appendix A.

Here, we augment the intuitive proof outlines we gave in the main body of the paper with more formal arguments. We start with a set of results which were used to classify graphs with non-identifiable effects. In the proofs presented here, we will construct the distributions which make up our set of premises to be positive. This is because non-positive distributions present a number of technical difficulties, for instance d-separation and independence are not related in a straightforward way in such distributions, and conditional distributions may not be defined. We should mention, however, that distributions which span multiple hypothetical worlds which we discussed in Section 4 may be non-positive by definition.

Theorem 5 *If sets $\boldsymbol{X}$ and $\boldsymbol{Y}$ are d-separated by $\boldsymbol{Z}$ in G, then $\boldsymbol{X}$ is independent of $\boldsymbol{Y}$ given $\boldsymbol{Z}$ in every P for which G is an I-map. Furthermore, the causal diagram induced by any semi-Markovian PCM M is a semi-Markovian I-map of the distribution $P(\boldsymbol{v},\boldsymbol{u})$ induced by M.*

Proof It is not difficult to see that if we restrict d-separation queries to a subset of variables $\mathbf{W}$ in some graph G, the corresponding independencies in $P(\mathbf{w})$ will only hold whenever the d-separation statements hold. Furthermore, if we replace G by a latent projection L (Pearl, 2000), where we view variables $\mathbf{V} \setminus \mathbf{W}$ as hidden, independencies in $P(\mathbf{w})$ will only hold whenever the corresponding d-separation statement (extended to include bidirected arcs) holds in L. ■

Theorem 10 *$P(\boldsymbol{v}), G \not\vdash_{id} P(y|do(x))$ in G shown in Fig. 1 (a).*

Proof We construct two causal models M^1 and M^2 such that $P^1(X,Y) = P^2(X,Y)$, and $P^1_x(Y) \neq P^2_x(Y)$. The two models agree on the following: all 3 variables are boolean, U is a fair coin, and $f_X(u) = u$. Let $\oplus$ denote the exclusive or (XOR) function. Then the value of Y is determined by the function $u \oplus x$ in M^1, while Y is set to 0 in M^2. Then $P^1(Y=0) = P^2(Y=0) = 1$, $P^1(X=0) = P^2(X=0) = 0.5$. Therefore, $P^1(X,Y) = P^2(X,Y)$, while $P^2_x(Y=0) = 1 \neq P^1_x(Y=0) = 0.5$. Note that while P is non-positive, it is straightforward to modify the proof for the positive case by letting f_Y functions in both models return 1 half the time, and the values outlined above half the time. ■

Theorem 12 *Let G be a Y-rooted C-tree. Let $\boldsymbol{X}$ be any subset of observable nodes in G which does not contain Y. Then $P(\boldsymbol{v}), G \not\vdash_{id} P(y|do(\boldsymbol{x}))$.*

Proof We generalize the proof for the bow arc graph. We can assume without loss of generality that each unobservable U in G has exactly two observable children. We construct two models with binary nodes. In the first model, the value of all observable nodes is set to the bit parity (sum modulo 2) of the parent values. In the second model, the same is true for all nodes except Y, with the latter being set to 0 explicitly. All $\mathbf{U}$ nodes in both models are fair coins. Since G is a tree, and since every $U \in \mathbf{U}$ has exactly two children in G, every $U \in \mathbf{U}$ has exactly two distinct downward paths to Y in G. It's then easy to establish that Y counts the bit parity of every node in $\mathbf{U}$ twice in the first model. But this implies $P^1(Y = 1) = 0$.

Because bidirected arcs form a spanning tree over observable nodes in G, for any set of nodes $\mathbf{X}$ such that $Y \notin \mathbf{X}$, there exists $U \in \mathbf{U}$ with one child in $An(\mathbf{X})_G$ and one child in $G \setminus An(\mathbf{X})_G$. Thus $P^1_{\mathbf{X}}(Y = 1) > 0$, but $P^2_{\mathbf{X}}(Y = 1) = 0$. It is straightforward to generalize this proof for the positive $P(\mathbf{v})$ in the same way as in Theorem 10. ■

Theorem 13 *$P(\boldsymbol{v}), G \not\vdash_{id} P(y|do(pa(y)))$ if and only if there exists a subgraph of G which is a Y-rooted C-tree.*

Proof From Tian (2002), we know that whenever there is no subgraph G' of G, such that all nodes in G' are ancestors of Y, and G' is a C-component, $P_{pa(Y)}(Y)$ is identifiable. From Theorem 12, we know that if there is a Y-rooted C-tree containing a non-empty subset S of parents of Y, then $P_s(Y)$ is not identifiable. But it is always possible to extend the counterexamples which prove non-identification of $P_s(Y)$ with additional variables which are independent. ■

Theorem 17 *Let F, F' be subgraphs of G which form a hedge for $P(\boldsymbol{y}|do(\boldsymbol{x}))$. Then $P(\boldsymbol{v}), G \not\vdash_{id} P(\boldsymbol{y}|do(\boldsymbol{x}))$.*

Proof We first show $P_{\mathbf{X}}(\mathbf{r})$ is not identifiable in F. As before, we assume each U has two observable children. We construct two models with binary nodes. In M^1 every variable in F is equal to the bit parity of its parents. In M^2 the same is true, except all nodes in F' disregard the parent values in $F \setminus F'$. All $\mathbf{U}$ are fair coins in both models.

As was the case with C-trees, for any C-forest F, every $U \in \mathbf{U} \cap F$ has exactly two downward paths to $\mathbf{R}$. It is now easy to establish that in M^1, $\mathbf{R}$ counts the bit parity of every node in $\mathbf{U}^1$ twice, while in M^2, $\mathbf{R}$ counts the bit parity of every node in $\mathbf{U}^2 \cap F'$ twice. Thus, in both models with no interventions, the bit parity of $\mathbf{R}$ is even.

Next, fix two distinct instantiations of $\mathbf{U}$ that differ by values of $\mathbf{U}^*$. Consider the topmost node $W \in F$ with an odd number of parents in $\mathbf{U}^*$ (which exists because bidirected edges in F form a spanning tree). Then flipping the values of $\mathbf{U}^*$ once will flip the value W once. Thus the function from $\mathbf{U}$ to $\mathbf{V}$ induced by a C-forest F in M^1 and M^2 is one to one.

The above results, coupled with the fact that in a C-forest, $|\mathbf{U}| + 1 = |\mathbf{V}|$ implies that any assignment where $\sum \mathbf{r} \pmod 2 = 0$ is equally likely, and all other node assignments are impossible in both F and F'. Since the two models agree on all functions and distributions in $F \setminus F'$, $\sum_{f'} P^1 = \sum_{f'} P^2$. It follows that the observational distributions are the same in both models.

As before, we can find $U \in \mathbf{U}$ with one child in $An(\mathbf{X})_F$, and one child in $F \setminus An(\mathbf{X})_F$, which implies the probability of odd bit parity of $\mathbf{R}$ is 0.5 in M^1, and 0 in M^2.

Next, we note that the construction so far results in a non-positive distribution P. To rid our proof of non-positivity, we "soften" our two models with new unobservable binary U_R for every $R \in \mathbf{R}$ which assumes value 1 with very small probability p. Whenever U_R is 1, the node R flips its value, otherwise it keeps the value as defined above. Note that $P(\mathbf{v})$ will remain the same in both models because our augmentation is the same, and the previous unsoftened models agreed on $P(\mathbf{v})$. It's easy to see that the bit parity of R in both models will be odd only when an odd number of U_R assume values of 1. Because p is arbitrarily small, the probability of an odd parity is far smaller than the probability of even parity. Now consider what happens after $do(\mathbf{x})$. In M^2, the probability of odd bit parity stays the same. In M^1 before the addition of U_R, the probability was 0.5. But it's easy to see that U_R nodes change the bit parity of $\mathbf{R}$ in a completely symmetric way, so the probability of even parity remains 0.5.

This implies $P_{\mathbf{x}}(\mathbf{r})$ is not identifiable. Finally, to see that $P_{\mathbf{x}}(\mathbf{y})$ is not identifiable, augment our counterexample by nodes in $\mathbf{I} = An(\mathbf{Y}) \cap De(\mathbf{R})$. Without loss of generality, assume every node in $\mathbf{I}$ has at most one child. Let each node I in $\mathbf{I}$ be equal to the bit parity of its parents. Moreover, each I has an exogenous parent U_I independent of the rest of $\mathbf{U}$ which, with small probability p causes it to flip it's value. Then the bit parity of $\mathbf{Y}$ is even if and only if an odd number of $\mathbf{U_I}$ turn on. Moreover, it's easy to see $P(\mathbf{I}|\mathbf{R})$ is positive by construction. We can now repeat the previous argument. ■

Next, we provide the proof of soundness of **ID** and **IDC** using do-calculus. This both simplifies the proofs and allows us to infer do-calculus is complete from completeness of our algorithms. We will invoke do-calculus rules by just using their number, for instance "by rule 2." First, we prove that a joint distribution in a causal model can be represented as a product of interventional distributions corresponding to the set of c-component in the graph induced by the model.

Lemma 33 (c-component factorization) *Let M be a causal model with graph G. Let $\boldsymbol{y}, \boldsymbol{x}$ be value assignments. Let $C(G \setminus \boldsymbol{X}) = \{S_1, ..., S_k\}$. Then $P_{\boldsymbol{x}}(\boldsymbol{y}) = \sum_{\boldsymbol{v} \setminus (\boldsymbol{y} \cup \boldsymbol{x})} \prod_i P_{\boldsymbol{v} \setminus s_i}(s_i)$.*

Proof A proof of this was derived by Tian (2002). Nevertheless, we reprove this result using do-calculus to help with our subsequent completeness results. Assume $\mathbf{X} = \emptyset$, $\mathbf{Y} = \mathbf{V}$, $C(G) = \{S_1, ..., S_k\}$, and let $A_i = An(S_i)_G \setminus S_i$. Then

$$\prod_i P_{\mathbf{v} \setminus s_i}(s_i) = \prod_i P_{a_i}(s_i) = \prod_i \prod_{V_j \in S_i} P_{a_i}(v_j | v_\pi^{(j-1)} \setminus a_i)$$

$$= \prod_i \prod_{V_j \in S_i} P(v_j | v_\pi^{(j-1)}) = \prod_i P(v_i | v_\pi^{(i-1)}) = P(\mathbf{v}).$$

The first identity is by rule 3, the second is by chain rule of probability. To prove the third identity, we consider two cases. If $A \in A_i \setminus V_\pi^{(j-1)}$, we can eliminate the intervention on A from the expression $P_{a_i}(v_j | v_\pi^{(j-1)})$ by rule 3, since $(V_j \perp\!\!\!\perp A | V_\pi^{(j-1)})_{G_{\overline{a_i}}}$.

If $A \in A_i \cap V_\pi^{(j-1)}$, consider any back-door path from A_i to V_j. Any such path with a node not in $V_\pi^{(j-1)}$ will be d-separated because, due to recursiveness, it must contain a blocked collider. Further, this path must contain bidirected arcs only, since all nodes on this path are conditioned or fixed.

Because $A_i \cap S_i = \emptyset$, all such paths are d-separated. The identity now follows from rule 2. The last two identities are just grouping of terms, and application of chain rule.

Having proven that c-component factorization holds for $P(\mathbf{v})$, we want to extend the result to $P_{\mathbf{x}}(\mathbf{y})$. First, let's consider $P_{\mathbf{x}}(\mathbf{v} \setminus \mathbf{x})$. This is just the distribution of the submodel $M_{\mathbf{x}}$. But $M_{\mathbf{x}}$ is just an ordinary causal model inducing $G \setminus \mathbf{X}$, so we can apply the same reasoning to obtain $P_{\mathbf{x}}(\mathbf{v} \setminus \mathbf{x}) = \prod_i P_{\mathbf{v} \setminus s_i}(s_i)$, where $C(G \setminus \mathbf{X}) = \{S_1, ..., S_k\}$. As a last step, it's easy to verify that $P_{\mathbf{x}}(\mathbf{y}) = \sum_{\mathbf{v} \setminus (\mathbf{x} \cup \mathbf{y})} P_{\mathbf{x}}(\mathbf{v} \setminus \mathbf{x})$. ■

Lemma 34 *Let $\boldsymbol{X}' = \boldsymbol{X} \cap An(\boldsymbol{Y})_G$. Then $P_{\boldsymbol{x}}(\boldsymbol{y})$ obtained from P in G is equal to $P'_{\boldsymbol{x}'}(\boldsymbol{y})$ obtained from $P' = P(An(\boldsymbol{Y}))$ in $An(\boldsymbol{Y})_G$.*

Proof Let $\mathbf{W} = \mathbf{V} \setminus An(\mathbf{Y})_G$. Then the submodel $M_{\mathbf{W}}$ induces the graph $G \setminus \mathbf{W} = An(\mathbf{Y})_G$, and its distribution is $P' = P_{\mathbf{W}}(An(\mathbf{Y})) = P(An(\mathbf{Y}))$ by rule 3. Now $P_{\mathbf{x}}(\mathbf{y}) = P_{\mathbf{x}'}(\mathbf{y}) = P_{\mathbf{x}', \mathbf{W}}(\mathbf{y}) = P'_{\mathbf{x}'}(\mathbf{y})$ by rule 3. ■

Lemma 35 *Let $\boldsymbol{W} = (\boldsymbol{V} \setminus \boldsymbol{X}) \setminus An(\boldsymbol{Y})_{G_{\overline{\boldsymbol{X}}}}$. Then $P_{\boldsymbol{x}}(\boldsymbol{y}) = P_{\boldsymbol{x}, \boldsymbol{w}}(\boldsymbol{y})$, where $\boldsymbol{w}$ are arbitrary values of $\boldsymbol{W}$.*

Proof Note that by assumption, $\mathbf{Y} \perp\!\!\!\perp \mathbf{W} | \mathbf{X}$ in $G_{\overline{\mathbf{X}}, \overline{\mathbf{W}}}$. The conclusion follows by rule 3. ■

Lemma 36 *When the conditions of line 6 are satisfied, $P_{\boldsymbol{x}}(\boldsymbol{y}) = \sum_{s \setminus \boldsymbol{y}} \prod_{V_i \in S} P(v_i | v_\pi^{(i-1)})$.*

Proof If line 6 preconditions are met, then G local to that recursive call is partitioned into S and $\mathbf{X}$, and there are no bidirected arcs from $\mathbf{X}$ to S. The conclusion now follows from the proof of Lemma 33. ■

Lemma 37 *Whenever the conditions of the last recursive call of **ID** are satisfied, $P_{\boldsymbol{x}}$ obtained from P in the graph G is equal to $P'_{\boldsymbol{x} \cap S'}$ obtained from $P' = \prod_{V_i \in S'} P(V_i | V_\pi^{(i-1)} \cap S', v_\pi^{(i-1)} \setminus S')$ in the graph S'.*

Proof It is easy to see that when the last recursive call executes, $\mathbf{X}$ and S partition G, and $\mathbf{X} \subset An(S)_G$. This implies that the submodel $M_{\mathbf{x} \setminus S'}$ induces the graph $G \setminus (\mathbf{X} \setminus S') = S'$. The distribution $P_{\mathbf{x} \setminus S'}$ of $M_{\mathbf{x} \setminus S'}$ is equal to P' by the proof of Lemma 33. It now follows that $P_{\mathbf{x}} = P_{\mathbf{x} \cap S', \mathbf{x} \setminus S'} = P'_{\mathbf{x} \cap S'}$. ■

Theorem 38 (soundness) *Whenever **ID** returns an expression for $P_{\boldsymbol{x}}(\boldsymbol{y})$, it is correct.*

Proof If $\mathbf{x} = \emptyset$, the desired effect can be obtained from P by marginalization, thus this base case is clearly correct. The soundness of all other lines except the failing line 5 has already been established. ■

Having established soundness, we show that whenever **ID** fails, we can recover a hedge for an effect involving a subset of variables involved in the original effect expression $P(\mathbf{y}|do(\mathbf{x}))$. This in turn implies completeness.

Theorem 39 *Assume **ID** fails to identify $P_{\boldsymbol{X}}(\boldsymbol{y})$ (executes line 5). Then there exist $\boldsymbol{X}' \subseteq \boldsymbol{X}$, $\boldsymbol{Y}' \subseteq \boldsymbol{Y}$ such that the graph pair G, S returned by the fail condition of **ID** contain as edge subgraphs C-forests F, F' that form a hedge for $P_{\boldsymbol{X}'}(\boldsymbol{y}')$.*

Proof Consider line 5, and G and $\mathbf{y}$ local to that recursive call. Let $\mathbf{R}$ be the root set of G. Since G is a single C-component, it is possible to remove a set of directed arrows from G while preserving the root set $\mathbf{R}$ such that the resulting graph F is an $\mathbf{R}$-rooted C-forest.

Moreover, since $F' = F \cap S$ is closed under descendants, and since only single directed arrows were removed from S to obtain F', F' is also a C-forest. $F' \cap \mathbf{X} = \emptyset$, and $F \cap \mathbf{X} \neq \emptyset$ by construction. $\mathbf{R} \subseteq An(\mathbf{Y})_{G_{\overline{\mathbf{X}}}}$ by lines 2 and 3 of the algorithm. It's also clear that $\mathbf{y}, \mathbf{x}$ local to the recursive call in question are subsets of the original input. ■

Theorem 18 ***ID** is complete.*

Proof By the previous theorem, if **ID** fails, then $P_{\mathbf{X}'}(\mathbf{y}')$ is not identifiable in a subgraph $H = G_{An(\mathbf{Y})_{\cap De(F)}}$ of G. Moreover, $\mathbf{X} \cap H = \mathbf{X}'$, by construction of H. As such, it is easy to extend the counterexamples in Theorem 39 with variables independent of H, with the resulting models inducing G, and witnessing the non-identifiability of $P_{\mathbf{X}}(\mathbf{y})$. ■

Next, we prove the results necessary to establish completeness of **IDC**.

Lemma 40 *If rule 2 of do-calculus applies to a set $\boldsymbol{Z}$ in G for $P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w})$ then there are no d-connected paths to $\boldsymbol{Y}$ that pass through $\boldsymbol{Z}$ in neither $G_1 = G \setminus \boldsymbol{X}$ given $\boldsymbol{Z}, \boldsymbol{W}$ nor in $G_2 = G \setminus (\boldsymbol{X} \cup \boldsymbol{Z})$ given $\boldsymbol{W}$.*

Proof Clearly, there are no d-connected paths through $\mathbf{Z}$ in G_2 given $\mathbf{W}$. Consider a d-connected path through $Z \in \mathbf{Z}$ to $\mathbf{Y}$ in G_1, given $\mathbf{Z}, \mathbf{W}$. Note that this path must either form a collider at Z or a collider which is an ancestor of Z. But this must mean there is a back-door path from $\mathbf{Z}$ to $\mathbf{Y}$, which is impossible, since rule 2 is applicable to $\mathbf{Z}$ in G for $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$. Contradiction. ■

Theorem 20 *For any G and any conditional effect $P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w})$ there exists a unique maximal set $\boldsymbol{Z} = \{Z \in \boldsymbol{W} | P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w}) = P_{\boldsymbol{X},z}(\boldsymbol{y}|\boldsymbol{w} \setminus \{z\})\}$ such that rule 2 applies to $\boldsymbol{Z}$ in G for $P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w})$. In other words, $P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w}) = P_{\boldsymbol{X},\boldsymbol{Z}}(\boldsymbol{y}|\boldsymbol{w} \setminus \boldsymbol{z})$.*

Proof Fix two maximal sets $\mathbf{Z}_1, \mathbf{Z}_2 \subseteq \mathbf{W}$ such that rule 2 applies to $\mathbf{Z}_1, \mathbf{Z}_2$ in G for $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$. If $\mathbf{Z}_1 \neq \mathbf{Z}_2$, fix $Z \in \mathbf{Z}_1 \setminus \mathbf{Z}_2$. By Lemma 40, rule 2 applies for $\{Z\} \cup \mathbf{Z}_2$ in G for $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$, contradicting our assumption.

Thus if we fix G and $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$, any set to which rule 2 applies must be a subset of the unique maximal set $\mathbf{Z}$. It follows that $\mathbf{Z} = \{Z \in \mathbf{W} | P_{\mathbf{X}}(\mathbf{y}|\mathbf{w}) = P_{\mathbf{X},z}(\mathbf{y}|\mathbf{w} \setminus \{z\})\}$. ■

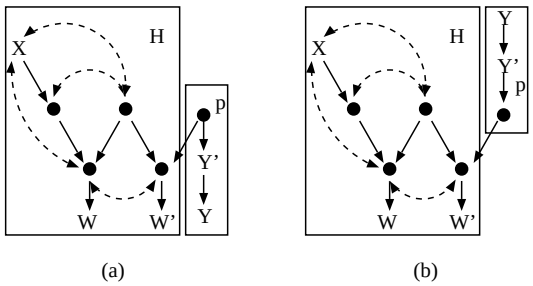


Figure 14: Inductive cases for proving non-identifiability of $P_x(y|w,w')$.

Lemma 41 *Let F, F' form a hedge for $P_{\boldsymbol{X}}(\boldsymbol{y})$. Then $F \subseteq F' \cup \boldsymbol{X}$.*

Proof It has been shown that **ID** fails on $P_{\mathbf{X}}(\mathbf{y})$ in G and returns a hedge if and only if $P_{\mathbf{X}}(\mathbf{y})$ is not identifiable in G. In particular, edge subgraphs of the graphs G and S returned by line 5 of **ID** form the C-forests of the hedge in question. It is easy to check that a subset of $\mathbf{X}$ and S partition G. ■

We rephrase the statement of Theorem 21 somewhat, to reduce "algebraic clutter."

Theorem 21 *Let $P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w})$ be such that every $W \in \boldsymbol{W}$ has a back-door path to $\boldsymbol{Y}$ in $G \setminus \boldsymbol{X}$ given $\boldsymbol{W} \setminus \{W\}$. Then $P_{\boldsymbol{X}}(\boldsymbol{y}|\boldsymbol{w})$ is identifiable in G if and only if $P_{\boldsymbol{X}}(\boldsymbol{y},\boldsymbol{w})$ is identifiable in G.*

Proof If $P_{\mathbf{X}}(\mathbf{y},\mathbf{w})$ is identifiable in G, then we can certainly identify $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$ by marginalization and division. The difficult part is to prove that if $P_{\mathbf{X}}(\mathbf{y},\mathbf{w})$ is not identifiable then neither is $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$.

Assume $P_{\mathbf{X}}(\mathbf{w})$ is identifiable. Then if $P_{\mathbf{X}}(\mathbf{y}|\mathbf{w})$ were identifiable, we would be able to compute $P_{\mathbf{X}}(\mathbf{y},\mathbf{w})$ by the chain rule. Thus our conclusion follows.

Assume $P_{\mathbf{X}}(\mathbf{w})$ is not identifiable. We also know that every $W \in \mathbf{W}$ contains a back-door path to some $Y \in \mathbf{Y}$ in $G \setminus \mathbf{X}$ given $\mathbf{W} \setminus \{W\}$. Fix such W and Y, along with a subgraph p of G which forms the witnessing back-door path. Consider also the hedge F, F' which witnesses the non-identifiability of $P_{\mathbf{X}'}(\mathbf{w}')$, where $\mathbf{X}' \subseteq \mathbf{X}, \mathbf{W}' \subseteq \mathbf{W}$.

Let $H = G_{De(F)\cup An(\mathbf{W}')_{G_{\overline{\mathbf{X}'}}}}$. We will attempt to show that $P_{\mathbf{X}'}(Y|\mathbf{w})$ is not identifiable in $H \cup p$. Without loss of generality, we make the following three assumptions. First, we restrict our attention to $\mathbf{W}'' \subseteq \mathbf{W}$ that occurs in $H \cup p$. Second, we assume p is a path segment which starts at H and ends at Y, and does not intersect H. Third, we assume all observable nodes in H have at most one child.

Consider the models M^1, M^2 from the proof of Theorem 17 which induce H. We extend the models by adding to them binary variables in p. Each variable $X \in p$ is equal to the bit parity of its parents, if it has any. If not, X behaves as a fair coin. If $Y \in H$ has a parent $X \in p$, the value of X is added to the bit parity computation Y makes.

Call the resulting models M^1_*, M^2_*. Because M^1, M^2 agreed on $P(H)$, and variables and functions in p are the same in both models, $P^1_* = P^2_*$. We will assume $\mathbf{w}''$ assigns 0 to every variable in $\mathbf{W}''$. What remains to be shown is that $P^1_{*\mathbf{X}}(y|\mathbf{w}'') \neq P^2_{*\mathbf{X}}(y|\mathbf{w}'')$. We will prove this by induction on the path structure of p. We handle the inductive cases first. In all these cases, we fix a node Y' that is between Y and H on the path p, and prove that if $P_{\mathbf{X}'}(y'|\mathbf{w}'')$ is not identifiable, then neither is $P_{\mathbf{X}'}(y|\mathbf{w}'')$.

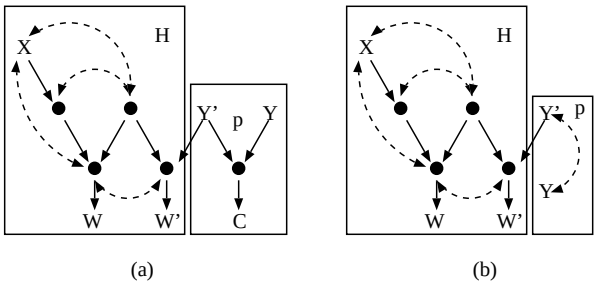


Figure 15: Inductive cases for proving non-identifiability of $P_x(y|w,w')$.

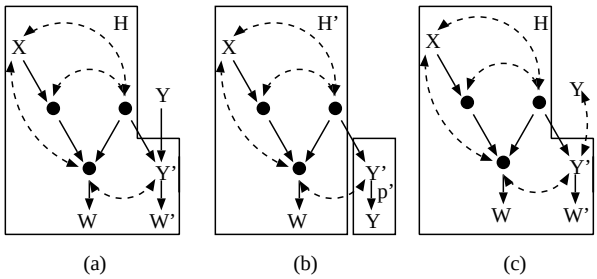


Figure 16: Base cases for proving non-identifiability of $P_x(y|w,w')$.

Assume neither Y nor Y' have descendants in $\mathbf{W}''$. If Y' is a parent of Y as in Fig. 14 (a), then $P_{\mathbf{X}'}(y|\mathbf{w}'') = \sum_{y'} P(y|y')P_{\mathbf{X}'}(y'|\mathbf{w}'')$. If Y is a parent of Y', as in Fig. 14 (b) then the next node in p must be a child of Y'. Therefore, $P_{\mathbf{X}'}(y|\mathbf{w}'') = \sum_{y'} P(y|y')P_{\mathbf{X}'}(y'|\mathbf{w}'')$. In either case, by construction $P(Y|Y')$ is a 2 by 2 identity matrix. This implies that the mapping from $P_{\mathbf{X}'}(y'|\mathbf{w}'')$ to $P_{\mathbf{X}'}(y|\mathbf{w}'')$ is one to one. If Y' and Y share a hidden common parent U as in Fig. 15 (b), then our result follows by combining the previous two cases.

The next case is if Y and Y have a common child C which is either in $\mathbf{W}''$ or has a descendant in $\mathbf{W}''$, as in Fig. 15 (a). Now $P_{\mathbf{X}'}(y|\mathbf{w}'') = \sum_{y'} P(y|y',c)P_{\mathbf{X}'}(y'|\mathbf{w}'')$. Because all nodes in $\mathbf{W}''$ were observed to be 0, $P(y|y',c)$ is again a 2 by 2 identity matrix.

Finally, we handle the base cases of our induction. In all such cases, Y is the first node not in H on the path p. Let Y' be the last node in H on the path p.

Assume Y is a parent of Y', as shown in Fig. 16 (a). By Lemma 41, we can assume $Y \notin An(F \setminus F')_H$. By construction, $(\sum \mathbf{W}'' = Y + 2 * \sum \mathbf{U}) \pmod 2$ in M^1_*, and $(\sum \mathbf{W}'' = Y + 2 * \sum(\mathbf{U} \cap F')) \pmod 2$ in M^2_*. If every variable in $\mathbf{W}''$ is observed to be 0, then $Y = (2 * \sum \mathbf{U}) \pmod 2$ in M^1_*, and $Y = (2 * \sum(\mathbf{U} \cap F')) \pmod 2$ in M^2_*. If an intervention $do(\mathbf{x})$ is performed, $(\sum \mathbf{W}'' = Y + 2 * \sum(\mathbf{U} \cap F')) \pmod 2$ in $M^2_{*\mathbf{X}}$, by construction. Thus if $\mathbf{W}''$ are all observed to be zero, $Y = 0$ with probability 1. Note that in $M^1_{\mathbf{X}}$ as constructed in the proof of Theorem 17, $(\sum \mathbf{w}'' = \mathbf{x} + \sum \mathbf{U}') \pmod 2$, where $\mathbf{U}' \subseteq \mathbf{U}$ consists of unobservable nodes with one child in $An(\mathbf{X})_F$ and one child in $F \setminus An(\mathbf{X})_F$.

Because $Y \notin An(F \setminus F')_H$, we can conclude that if $\mathbf{W}''$ are observed to be 0, $Y = (\mathbf{x} + \sum \mathbf{U}') \pmod 2$ in $M^1_{*\mathbf{X}'}$. Thus, $Y = 0$ with probability 0.5. Therefore, $P^1_{*\mathbf{X}'}(y|\mathbf{w}'') \neq P^2_{*\mathbf{X}'}(y|\mathbf{w}'')$ in this case.

Assume Y is a child of Y'. Now consider a graph G' which is obtained from $H \cup p$ by removing the (unique) outgoing arrow from Y' in H. If $P_{\mathbf{X}'}(Y|\mathbf{w}'')$ is not identifiable in G', we are done.

Assume $P_{\mathbf{X}'}(Y|\mathbf{w}'')$ is identifiable in G'. If $Y' \in F$, and $\mathbf{R}$ is the root set of F, then removing the Y'-outgoing directed arrow from F results in a new C-forest, with a root set $\mathbf{R} \cup \{Y'\}$. Because Y is a child of Y', the new C-forests form a hedge for $P_{\mathbf{X}'}(y, \mathbf{w}'')$. If $Y' \in H \setminus F$, then removing the Y'-outgoing directed arrow results in substituting Y for $W \in \mathbf{W}'' \cap De(Y')_H$. Thus in G', F, F' form a hedge for $P_{\mathbf{X}'}(y, \mathbf{w}'' \setminus \{w\})$. In either case, $P_{\mathbf{X}'}(y, \mathbf{w}'')$ is not identifiable in G'.

If $P_{\mathbf{X}'}(\mathbf{w}'')$ is identifiable in G', we are done. If not, consider a smaller hedge $H' \subset H$ witnessing this fact. Now consider the segment p' of p between Y and H'. We can repeat the inductive argument for H', p' and Y. See Fig. 16 (b).

If $P_{\mathbf{X}'}(\mathbf{w}'')$ is identifiable in G', we are done. If not, consider a smaller hedge $H' \subset H$ witnessing this fact. Now consider the segment p' of p between Y and H'. We can repeat the inductive argument for H', p' and Y. See Fig. 16 (b). If Y and Y' have a hidden common parent, as is the case in Fig. 16 (c), we can combine the first inductive case, and the first base case to prove our result.

We conclude the proof by introducing a slight change to rid us of non-positivity in the distributions P^1, P^2 in our counterexamples. Specifically, for every node I in $p \cup (De(\mathbf{R}) \cap An(\mathbf{Y}))$, add a new binary exogenous parent U_I which is independent of other nodes in $\mathbf{U}$, and has an arbitrarily small probability of assuming the value 1, and causing its child to flip its current value. We let P_{odd} be the probability an odd number of U_I nodes assume the value 1. Because $P(U_I = 1)$ is vanishingly small for every I, P_{odd} is much smaller than 0.5. It's easy to see that P is positive in counterexamples augmented in this way. In the base case when Y is a parent of Y', we modify our equations to account for the addition of U_I. Specifically, $(\sum \mathbf{W}'' = Y + 2 * \sum \mathbf{U} + \sum \mathbf{U_I} \pmod 2)$ in M^1_*, and $(\sum \mathbf{W}'' = Y + 2 * \sum (\mathbf{U} \cap F') + \sum \mathbf{U_I} \pmod 2)$ in M^2_*, where $U_{\mathbf{U}}$ is the set of nodes added. If every variable in $\mathbf{W}''$ is observed to be 0, then $Y = (2 * \sum \mathbf{U} + \sum \mathbf{U_I} \pmod 2)$ in M^1_*, and $Y = (2 * \sum (\mathbf{U} \cap F') + \sum \mathbf{U_I} \pmod 2)$ in M^2_*. So prior to the intervention, $P(Y = 1|\mathbf{w}'') = P_{odd}$. But because $P^1_{\mathbf{X}'}(Y = 1|\mathbf{w}'') = 0.5$, adding U_I nodes to the model does not change this probability. Because $P^2(Y = 1|\mathbf{w}'') = P^2_{\mathbf{X}}(Y = 1|\mathbf{w}'')$, our conclusion follows.

In the inductive cases above, we showed that $P_{\mathbf{X}}(Y' = Y|\mathbf{W}'') = 1$ in our counterexamples. It's easy to see that with the addition of U_I, $P_{\mathbf{X}}(Y' = Y|\mathbf{W}'') = P_{odd}$. This implies that if $P^1_{\mathbf{X}}(Y'|\mathbf{W}'') \neq P^2_{\mathbf{X}}(Y'|\mathbf{W}'')$, then $P^1_{\mathbf{X}}(Y|\mathbf{W}'') \neq P^2_{\mathbf{X}}(Y|\mathbf{W}'')$.

This completes the proof. ■

What remains for us to show are the theorems which imply the soundness and completeness results in Section 4. The most important point in these proofs is that counterfactual graphs are generally no different from causal diagrams discussed in Sections 2 and 3, with their only special feature being that by construction, some nodes in the graph happen to share functions. This means that a lot of results we already proved for Section 3 can be reused without change.

Lemma 42 *If the preconditions of line 7 are met,* $P(S) = P_{\boldsymbol{x}}(\boldsymbol{var}(S))$, *where* $\boldsymbol{x} = \bigcup \boldsymbol{sub}(S)$.

Proof Let $\mathbf{x} = \bigcup \mathbf{sub}(S)$. Since the preconditions are met, $\mathbf{x}$ does not contain conflicting assignments to the same variable, which means $do(\mathbf{x})$ is a sound action in the original causal model. Note that for any variable $Y_{\mathbf{W}}$ in S, any variable in $(Pa(S) \setminus S) \cap An(Y_{\mathbf{W}})_S$ is already in $\mathbf{w}$, while any variable in $(Pa(S) \setminus S) \setminus An(Y_{\mathbf{W}})_S$ can be added to the subscript of $Y_{\mathbf{W}}$ without changing the variable. Since $Y \cap \mathbf{X} = \emptyset$ by assumption, $Y_{\mathbf{W}} = Y_{\mathbf{X}}$. Since $Y_{\mathbf{W}}$ was arbitrary, our result follows. ■

For convenience, we show the soundness of **ID*** and **IDC*** asserted in Theorem 26 separately.

Theorem 26 (a) *If **ID*** succeeds, the expression it returns is equal to $P(\gamma)$ in a given causal graph.*

Proof The proof outline in Section 3 is sufficient for everything except the base cases. In particular, line 6 follows by Lemma 33. For soundness, we only need to handle the positive base case, which follows from Lemma 42. ■

The soundness of **IDC*** is also fairly straightforward.

Theorem 26 (b) *If **IDC*** does not output **FAIL**, the expression it returns is equal to $P(\gamma|\delta)$ in a given causal graph, if that expression is defined, and **UNDEFINED** otherwise.*

Proof Theorem 20 shows how an operation similar to line 4 is sound by rule 2 of do-calculus (Pearl, 1995) when applied in a causal diagram. But we know that the counterfactual graph is just a causal diagram for a model where some nodes share functions, so the same reasoning applies. The rest is straightforward. ■

To show completeness of **ID*** and **IDC***, we first prove a utility lemma which will make it easier to construct counterexamples which agree on P_* but disagree on a given counterfactual query.

Lemma 43 *Let G be a causal graph partitioned into a set $\{S_1,...,S_k\}$ of C-components. Then two models M_1, M_2 which induce G agree on P_* if and only if their submodels $M^1_{\mathbf{v}\setminus s_i}$, $M^2_{\mathbf{v}\setminus s_i}$ agree on P_* for every C-component S_i, and value assignment $\mathbf{v}\setminus s_i$.*

Proof This follows from C-component factorization: $P(\mathbf{v}) = \prod_i P_{\mathbf{V}\setminus s_i}(s_i)$. This implies that for every $do(\mathbf{x})$, $P_{\mathbf{X}}(\mathbf{v})$ can be expressed as a product of terms $P_{\mathbf{V}\setminus(s_i\setminus\mathbf{X})}(s_i\setminus\mathbf{x})$, which implies the result. ■

The next result generalizes Lemma 27 to a wider set of counterfactual graphs which result from non-identifiable queries.

Lemma 28 *Assume G is such that X is a parent of Y and Z, and Y and Z are connected by a bidirected path with observable nodes $W^1,...,W^k$ on the path. Then $P_*, G \not\vdash_{id} P(y_x, w^1,...,w^k, z_{x'})$, $P(y_x, w^1,...,w^k, z)$ for any value assignments $y, w^1,...,w^k, z$.*

Proof We construct two models with graph G as follows. In both models, all variables are binary, and $P(\mathbf{u})$ is uniform. In M^1, each variable is set to the bit parity of its parents. In M^2, the same is true except Y and Z ignore the values of X. To prove that the two models agree on P_*, we use Lemma 43. Clearly the two models agree on $P(X)$. To show that the models also agree on $P_x(\mathbf{V}\setminus X)$ for all values of x, note that in M_2 each value assignment over $\mathbf{V}\setminus X$ with even bit parity is equally likely, while no assignment with odd bit parity is possible. But the same is true in M^1 because any value of x contributes to the bit parity of $\mathbf{V}\setminus X$ exactly twice. The agreement of M^1_x, M^2_x on P_* follows by the graph structure of G.

To see that the result is true, we note firstly that $P(\Sigma_i W^i + Y_x + Z_{x'} \pmod 2 = 1) = P(\Sigma_i W^i + Y_x + Z \pmod 2 = 1) = 0$ in M^2, while the same probabilities are positive in M^1, and secondly that in both models distributions $P(y_x, w^1,...,w^k, z_{x'})$ and $P(y_x, w^1,..,w^k, z)$ assign equal probabilities to

outcomes with positive probabilities, while we just established that the set of these possible outcomes differs in M_1 and M_2. Note that the proof is easy to generalize for positive P_* by adding a small probability for Y to flip its normal value. ■

To obtain a full characterization of non-identifiable counterfactual graphs, we augment the difficult graphs we obtained from the previous two results using certain graph transformation rules which preserve non-identifiability. These rules are given in the following two lemmas.

Lemma 29 *Assume $P_*, G \not\vdash_{id} P(\gamma)$. Let $\{y^1_{\mathbf{x}^1}, ..., y^n_{\mathbf{x}^m}\}$ be a subset of counterfactual events in γ. Let G' be a graph obtained from G by adding a new child W of $Y^1, ..., Y^n$, and let P'_* be the set of all interventional distributions in models inducing G'. Let $\gamma' = (\gamma \setminus \{y^1_{\mathbf{x}^1}, ..., y^n_{\mathbf{x}^m}\}) \cup \{w_{\mathbf{x}^1}, ..., w_{\mathbf{x}^m}\}$, where w is an arbitrary value of W. Then $P'_*, G' \not\vdash_{id} P(\gamma')$.*

Proof Let M^1, M^2 witness $P_*, G \not\vdash_{id} P(\gamma)$. We will extend these models to witness $P'_*, G' \not\vdash_{id} P(\gamma')$. Since the function of a newly added W will be shared, and M^1, M^2 agree on P_* in G, the extensions will agree on P'_* by Lemma 43. We have two cases.

Assume there is a variable Y^i such that $y^i_{\mathbf{x}^j}, y^i_{\mathbf{x}^k}$ are in γ. By Lemma 27, $P_*, G \not\vdash_{id} P(y^i_{\mathbf{x}^j}, y^i_{\mathbf{x}^k})$. Then let W be a child of just Y^i, and assume $|W| = |Y^i| = c$. Let W be set to the value of Y^i with probability $1-\varepsilon$, and otherwise it is set to a uniformly chosen random value of Y^i among the other $c-1$ values. Since ε is arbitrarily small, and since $W_{\mathbf{x}^j}$ and $W_{\mathbf{x}^k}$ pay attention to the same U variable, it is possible to set ε in such a way that if $P^1(Y^i_{\mathbf{x}^j}, Y^i_{\mathbf{x}^k}) \neq P^2(Y^i_{\mathbf{x}^j}, Y^i_{\mathbf{x}^k})$, however minutely, then $P^1(W_{\mathbf{x}^j}, W_{\mathbf{x}^k}) \neq P^2(W_{\mathbf{x}^j}, W_{\mathbf{x}^k})$.

Otherwise, let $|W| = \prod_i |Y^i|$, and let $P(W|Y^1, ..., Y^n)$ be an invertible stochastic matrix. Our result follows. ■

Lemma 30 *Assume $P_*, G \not\vdash_{id} P(\gamma)$. Let G' be obtained from G by merging some two nodes X, Y into a new node Z where Z inherits all the parents and children of X, Y, subject to the following restrictions:*

- *The merge does not create cycles.*
- *If $(\exists w_{\mathbf{s}} \in \gamma)$ where $x \in \mathbf{s}$, $y \notin \mathbf{s}$, and $X \in An(W)_G$, then $Y \notin An(W)_G$.*
- *If $(\exists y_{\mathbf{s}} \in \gamma)$ where $x \in \mathbf{s}$, then $An(X)_G = \emptyset$.*
- *If $(Y_{\mathbf{w}}, X_{\mathbf{s}} \in \gamma)$, then $\mathbf{w}$ and $\mathbf{s}$ agree on all variable settings.*

Assume $|X| \times |Y| = |Z|$ and there's some isomorphism f assigning value pairs x, y to a value $f(x,y) = z$. Let γ' be obtained from γ as follows. For any $w_{\mathbf{s}} \in \gamma$:

- *If $W \notin \{X, Y\}$, and values x, y occur in $\mathbf{s}$, replace them by $f(x,y)$.*
- *If $W \notin \{X, Y\}$, and the value of one of X, Y occur in $\mathbf{s}$, replace it by some z consistent with the value of X or Y.*
- *If X, Y do not occur in γ, leave γ as is.*
- *If $W = Y$ and $x \in \mathbf{s}$, replace $w_{\mathbf{s}}$ by $f(x,y)_{\mathbf{s} \setminus \{x\}}$.*

- *otherwise, replace every variable pair of the form* $Y_{\mathbf{r}} = y, X_{\mathbf{s}} = x$ *by* $Z_{\mathbf{r},\mathbf{s}} = f(x,y)$.

Then $P_*, G' \not\vdash_{id} P(\gamma')$.

Proof Let Z be the Cartesian product of X, Y, and fix f. We want to show that the proof of non-identification of $P(\gamma)$ in G carries over to $P(\gamma')$ in G'.

We have five modification conditions which can apply to a variable $w_{\mathbf{s}} \in \gamma$. However, since γ is left alone if X, Y do not occur in γ (the third condition), only the remaining four of these conditions result in an actual modification of a counterfactual variable in γ.

We go through these remaining conditions one by one. The first clearly results in the same counterfactual variable. For the second, due to the restrictions we imposed, $w_{\mathbf{z}} = w_{\mathbf{z},y,x}$, which means we can apply the first modification.

For the fourth, we have $P(\gamma) = P(\delta, y_{x,\mathbf{z}})$. By our restrictions, and rule 2 of do-calculus (Pearl, 1995), this is equal to $P(\delta, y_{\mathbf{z}} | x_{\mathbf{z}})$. Since this is not identifiable, then neither is $P(\delta, y_{\mathbf{z}}, x_{\mathbf{z}})$. Now it's clear that our modification is equivalent to one applied after the fifth condition.

The fifth modification is simply a merge of events consistent with a single causal world into a conjunctive event, which does not change the overall expression. ■

We are now ready to show the main completeness results for counterfactual identification algorithms. Again, we prove this results separately for **ID*** and **IDC*** for convenience.

Theorem 31 (a) ***ID**** *is complete.*

Proof We want to show that if line 8 fails, the original $P(\gamma)$ cannot be identified. There are two broad cases to consider. If G_γ contains the w-graph, the result follows by Lemmas 27 and 29. If not, we argue as follows.

Fix some X which witnesses the precondition on line 8. We can assume X is a parent of some nodes in S. Assume no other node in $\mathbf{sub}(S)$ affects S (effectively we delete all edges from parents of S to S except from X). Because the w-graph is not a part of G_γ, this has no ramifications on edges in S. Further, we assume X has two values in S.

If $X \notin S$, fix $Y, W \in S \cap Ch(X)$. Assume S has no directed edges at all. Then $P_*, G \not\vdash_{id} P(S)$ by Lemma 28. The result now follows by Lemma 29, and by construction of G_γ, which implies all nodes in S have some descendant in γ.

If S has directed edges, we want to show $P_*, G \not\vdash_{id} P(R(S))$, where $R(S)$ is the subset of S with no children in S. We can recover this from the previous case as follows. Assume S has no edges as before. For a node $Y \in S$, fix a set of childless nodes $\mathbf{X} \in S$ which are to be their parents. Add a virtual node Y' which is a child of all nodes in $\mathbf{X}$. Then $P_*, G \not\vdash_{id} P((S \setminus \mathbf{X}) \cup Y')$ by Lemma 29. Then $P_*, G \not\vdash_{id} P(R(S'))$, where S' is obtained from S by adding edges from $\mathbf{X}$ to Y by Lemma 30, which applies because no w-graph exists in G_γ. We can apply this step inductively to obtain the desired forest (all nodes have at most one child) S while making sure $P_*, G \not\vdash_{id} P(R(S))$.

If S is not a forest, we can simply disregard extra edges so effectively it is a forest. Since the w-graph is not in G_γ this does not affect edges from X to S.

If $X \in S$, fix $Y \in S \cap Ch(X)$. If S has no directed edges at all, replace X by a new virtual node Y, and make X be the parent of Y. By Lemma 28, $P_*, G \not\vdash_{id} P((S \setminus x) \cup y_x)$. We now repeat the same steps as before, to obtain that $P_*, G \not\vdash_{id} P((R(S) \setminus x) \cup y_x)$ for general S. Now we use Lemma 30 to

obtain $P_*, G \not\vdash_{id} P(R(S))$. Having shown $P_*, G \not\vdash_{id} P(R(S))$, we conclude our result by inductively applying Lemma 29. ■

Theorem 31 (b) ***IDC**** *is complete.*

Proof The difficult step is to show that after line 5 is reached, if $P_*, G \not\vdash_{id} P(\gamma, \delta)$ then $P_*, G \not\vdash_{id} P(\gamma|\delta)$. If $P_*, G \vdash_{id} P(\delta)$, this is obvious. Assume $P_*, G \not\vdash_{id} P(\delta)$. Fix the S which witnesses that for $\delta' \subseteq \delta$, $P_*, G \not\vdash_{id} P(\delta')$. Fix some Y such that a back-door, that is, starting with an incoming arrow, path exists from δ' to Y in $G_{\gamma,\delta}$. We want to show that $P_*, G \not\vdash_{id} P(Y|\delta')$. Let $G' = G_{An(\delta') \cap De(S)}$.

Assume Y is a parent of a node $D \in \delta'$, and $D \in G'$. Augment the counterexample models which induce counterfactual graph G' with an additional binary node for Y, and let the value of D be set as the old value plus Y modulo $|D|$. Let Y attain value 1 with vanishing probability ε. That the new models agree on P_* is easy to establish. To see that $P_*, G \not\vdash_{id} P(\delta')$ in the new model, note that $P(\delta')$ in the new model is equal to $P(\delta' \setminus D, D = d) * (1-\varepsilon) + P(\delta' \setminus D, D = (d-1) \pmod{|D|}) * \varepsilon$. Because ε is arbitrarily small, this implies our result. To show that $P_*, G \not\vdash_{id} P(Y = 1|\delta')$, we must show that the models disagree on $P(\delta'|Y = 1)/P(\delta')$. But to do this, we must simply find two consecutive values of D, $d, d+1 \pmod{|D|}$ such that $P(\delta' \setminus D, d+1 \pmod{|D|})/P(\delta' \setminus D, d)$ is different in the two models. But this follows from non-identification of $P(\delta')$.

If Y is not a parent of $D \in G'$, then either it is further along on the back-door path or it's a child of some node in G'. In case 1, we must construct the distributions along the back-door path in such a way that if $P_*, G \not\vdash_{id} P(Y'|\delta')$ then $P_*, G \not\vdash_{id} P(Y|\delta')$, where Y' is a node preceding Y on the path. The proof follows closely the one in Theorem 21. In case 2, we duplicate the nodes in G' which lead from Y to δ', and note that we can show non-identification in the resulting graph using reasoning in case 1. We obtain our result by applying Lemma 30. ■

References

Chen Avin, Ilya Shpitser, and Judea Pearl. Identifiability of path-specific effects. In *International Joint Conference on Artificial Intelligence*, volume 19, pages 357–363, 2005.

Alexander Balke and Judea Pearl. Counterfactual probabilities: Computational methods, bounds and applications. In *Proceedings of UAI-94*, pages 46–54, 1994a.

Alexander Balke and Judea Pearl. Probabilistic evaluation of counterfactual queries. In *Proceedings of AAAI-94*, pages 230–237, 1994b.

Alexander Philip Dawid. Conditional independence in statistical theory. *Journal of the Royal Statistical Society*, 41:1–31, 1979.

David Galles and Judea Pearl. An axiomatic characterization of causal counterfactuals. *Foundation of Science*, 3:151–182, 1998.

Trygve Haavelmo. The statistical implications of a system of simultaneous equations. *Econometrica*, 11: 1–12, 1943.

Joseph Halpern. Axiomatizing causal reasoning. *Journal of A.I. Research*, pages 317–337, 2000.

Yimin Huang and Marco Valtorta. Pearl's calculus of interventions is complete. In *Twenty Second Conference On Uncertainty in Artificial Intelligence*, 2006a.

Yimin Huang and Marco Valtorta. Identifiability in causal bayesian networks: A sound and complete algorithm. In *Twenty-First National Conference on Artificial Intelligence*, 2006b.

Manabu Kuroki and Masami Miyakawa. Identifiability criteria for causal effects of joint interventions. *Journal of Japan Statistical Society*, 29:105–117, 1999.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000. ISBN 0-521-77362-8.

Judea Pearl. Direct and indirect effects. In *Proceedings of UAI-01*, pages 411–420, 2001.

Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan and Kaufmann, San Mateo, 1988.

Judea Pearl. Graphical models, causality, and intervention. *Statistical Science*, 8:266–9, 1993a.

Judea Pearl. A probabilistic calculus of actions. In *Uncertainty in Artificial Intelligence (UAI)*, volume 10, pages 454–462, 1993b.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–709, 1995. URL `citeseer.ist.psu.edu/55450.html`.

Judea Pearl and James M. Robins. Probabilistic evaluation of sequential plans from causal models with hidden variables. In *Uncertainty in Artificial Intelligence*, volume 11, pages 444–453, 1995.

Ilya Shpitser and Judea Pearl. Identification of joint interventional distributions in recursive semi-markovian causal models. In *Twenty-First National Conference on Artificial Intelligence*, 2006a.

Ilya Shpitser and Judea Pearl. Identification of conditional interventional distributions. In *Uncertainty in Artificial Intelligence*, volume 22, 2006b.

Ilya Shpitser and Judea Pearl. What counterfactuals can be tested. In *Twenty Third Conference on Uncertainty in Artificial Intelligence, forthcoming*. Morgan Kaufmann, 2007.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer Verlag, New York, 1993.

Jin Tian. *Studies in Causal Reasoning and Learning*. PhD thesis, Department of Computer Science, University of California, Los Angeles, 2002.

Thomas S. Verma. Causal networks: semantics and expressiveness. Technical Report R-65, Cognitive Systems Laborator, University of California, Los Angeles, 1986.

Sewall Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–585, 1921.

Journal of Machine Learning Research 9 (2008) 1981-2014 Submitted 5/07; Revised 6/08; Published 9/08

Mixed Membership Stochastic Blockmodels

Edoardo M. Airoldi[*] EAIROLDI@PRINCETON.EDU
David M. Blei BLEI@CS.PRINCETON.EDU
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA

Stephen E. Fienberg[†] FIENBERG@STAT.CMU.EDU
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Eric P. Xing EPXING@CS.CMU.EDU
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Editor: Tommi Jaakkola

Abstract

Consider data consisting of pairwise measurements, such as presence or absence of links between pairs of objects. These data arise, for instance, in the analysis of protein interactions and gene regulatory networks, collections of author-recipient email, and social networks. Analyzing pairwise measurements with probabilistic models requires special assumptions, since the usual independence or exchangeability assumptions no longer hold. Here we introduce a class of variance allocation models for pairwise measurements: mixed membership stochastic blockmodels. These models combine global parameters that instantiate dense patches of connectivity (blockmodel) with local parameters that instantiate node-specific variability in the connections (mixed membership). We develop a general variational inference algorithm for fast approximate posterior inference. We demonstrate the advantages of mixed membership stochastic blockmodels with applications to social networks and protein interaction networks.

Keywords: hierarchical Bayes, latent variables, mean-field approximation, statistical network analysis, social networks, protein interaction networks

1. Introduction

The problem of modeling relational information among objects, such as pairwise relations represented as graphs, arises in a number of settings in machine learning. For example, scientific literature connects papers by citations, the Web connects pages by links, and protein-protein interaction data connects proteins by physical binding records. In these settings, we often wish to infer hidden attributes of the objects from the observed measurements on pairwise properties. For example, we might want to compute a clustering of the web-pages, predict the functions of a protein, or assess

∗. Also in the Lewis-Sigler Institute for Integrative Genomics. Address correspondence to 228 Carl Icahn Laboratory, Princeton University.
†. Also in the School of Computer Science.

the degree of relevance of a scientific abstract to a scholar's query. Unlike traditional data collected from individual objects, *relational data* violate the classical independence or exchangeability assumptions made in machine learning and statistics. The observations are dependent because of the way they are connected. This interdependence suggests that a different set of assumptions is more appropriate.

There is a history of research devoted to analyzing relational data. One well-studied problem is *clustering*, grouping the objects to uncover a structure based on the observed patterns of interactions. Standard model-based clustering methods, for example, mixture models, are not immediately applicable to relational data because they assume that the objects are conditionally independent given their cluster assignments. Rather, the latent stochastic blockmodel (Wang and Wong, 1987; Snijders and Nowicki, 1997) is an adaptation of mixture modeling to relational data. In that model, each object belongs to a cluster and the relationships between objects are governed by the corresponding pair of clusters. With posterior inference, one identifies a set of latent roles which govern the objects relationships with each other. A recent extension of this model relaxed the finite-cardinality assumption on the latent clusters with a nonparametric hierarchical Bayesian model based on the Dirichlet process prior (Kemp et al., 2004, 2006; Xu et al., 2006).

The latent stochastic blockmodel suffers from a limitation that each object can only belong to one cluster, or in other words, play a single latent role. However, many relational data sets are multi-facet. For example, when a protein or a social actor interacts with different partners, different functional or social contexts may apply and thus the protein or the actor may be acting according to different latent roles they can possible play. In this paper, we relax the assumption of single-latent-role for actors, and develop a *mixed membership model* for relational data. Mixed membership models, such as latent Dirichlet allocation (Blei et al., 2003), have re-emerged in recent years as a flexible modeling tool for data where the single cluster assumption is violated by the heterogeneity within of a data point. For almost two decades, these models have been successfully applied in many domains, such as surveys (Berkman et al., 1989; Erosheva, 2002), population genetics (Pritchard et al., 2000), document analysis (Minka and Lafferty, 2002; Blei et al., 2003; Buntine and Jakulin, 2006), image processing (Li and Perona, 2005), and transcriptional regulation (Airoldi et al., 2007).

The mixed membership model associates each unit of observation with multiple clusters rather than a single cluster, via a membership probability-like vector. The concurrent membership of a data in different clusters can capture its different aspects, such as different underlying topics for words constituting each document. This is also a natural idea for relational data, where the objects can bear multiple latent roles or cluster-memberships that influence their relationships to others. As we will demonstrate, a mixed membership approach to relational data lets us describe the interaction between objects playing multiple roles. For example, some of a protein's interactions may be governed by one function; other interactions may be governed by another function.

Existing mixed membership models are not appropriate for relational data because they assume that the data are conditionally independent given their latent membership vectors. In relational data, where each object is described by its relationships to others, we would like to assume that the ensemble of mixed membership vectors help govern the relationships of each object. The conditional independence assumptions of modern mixed membership models do not apply.

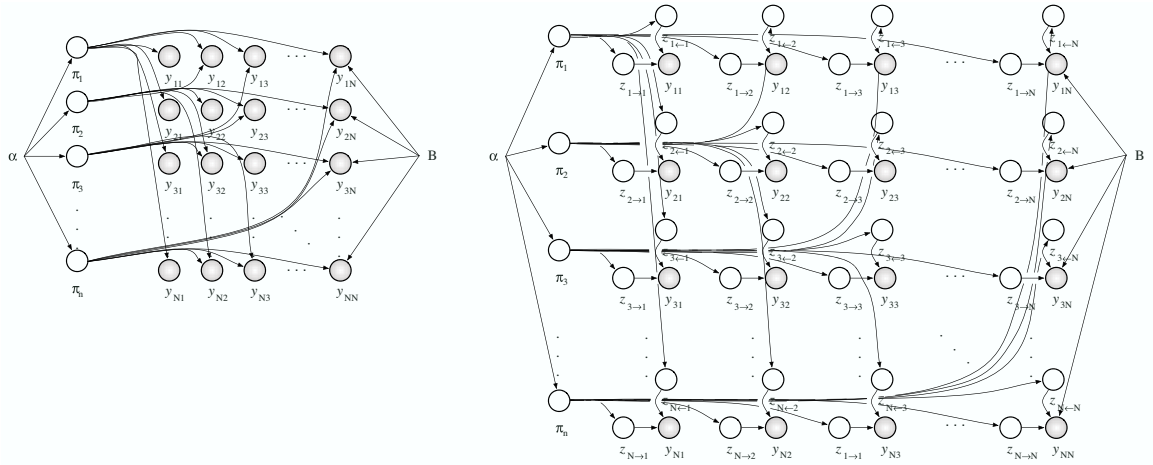


Figure 1: Two graphical model representations of the mixed membership stochastic blockmodel (MMB). Intuitively, the MMB summarized the variability of a graph with the blockmodel B and node-specific mixed membership vectors (left). In detail, a mixed membership, $\pi_n(k)$, quantifies the expected proportion of times node n instantiates the connectivity pattern of group k, according to the blockmodel. In any given interaction, $Y(n,m)$, however, node n instantiates the connectivity pattern of a single group, $z_{n\to m}(k)$. (right). We did not draw all the arrows out of the block model B for clarity; all interactions depend on it.

In this paper, we develop mixed membership models for relational data.[1] Models in this family include parameters to reduce bias due to sparsity, and can be used to analyze multiple collections of paired measurements, and collections of non-binary and multivariate paired measurements. We develop a fast nested variational inference algorithm that performs well in the relational setting and is parallelizable. We demonstrate the application of our technique to large-scale protein interaction networks and social networks. Our model captures the multiple roles that objects exhibit in interaction with others, and the relationships between those roles in determining the observed interaction matrix.

Mixed membership and the latent block structure can be recovered from relational data (Section 4.1). The application to a friendship network among students tests the model on a real data set where a well-defined latent block structure exists (Section 4.2). The application to a protein interaction network tests to what extent our model can reduce the dimensionality of the data, while revealing substantive information about the functionality of proteins that can be used to inform subsequent analyses (Section 4.3).

2. The Mixed Membership Stochastic Blockmodel

In this section, we describe the modeling assumptions if the mixed membership model of relational data. We represent observed relational data as a graph $G = (\mathcal{N}, Y)$, where $Y(p,q)$ maps pairs of nodes to values, that is, edge weights. We consider binary matrices, where $Y(p,q) \in \{0,1\}$. The data can be thought of as a directed graph.

As a running example, we consider the monk data of Sampson (1968). Sampson measured a collection of sociometric relations among a group of monks by repeatedly asking questions such as "whom do you like?" and "whom do you dislike?" to determine asymmetric social relationships within the group. The questionnaire was repeated at four subsequent epochs. Information about these repeated, asymmetric relations was collapsed into a square binary table that encodes the directed connections between monks by Breiger et al. (1975). In analyzing this data, the goal is to determine the social structure within the monastery.

In the context of the monastery example, we assume K factions, that is, latent groups, exist in the monastery, and the observed network is generated according to distributions of group-membership for each monk and a matrix of group-group interaction strength. The per-monk distributions are specified by latent simplicial vectors. Each monk is associated with a randomly drawn vector $\vec{\pi}_i$ for monk i, where $\pi_{i,g}$ denotes the probability of monk i belonging to group g. That is, each monk can simultaneously belong to multiple groups with different degrees of affiliation strength. The probabilities of interactions between different groups are defined by a matrix of Bernoulli rates $B_{(K\times K)}$, where $B(g,h)$ represents the probability of having a link between a monk from group g and a monk from group h.

For each monk, the indicator vector $\vec{z}_{p\to q}$ denotes the group membership of monk p when he responds to survey questions about monk q and $\vec{z}_{p\leftarrow q}$ denotes the group membership of monk q when he responds to survey questions about node p.[2] N denotes the number of monks in the monastery, and recall that K denotes the number of distinct groups a monk can belong to.

More in general, monks can be represented by nodes in a graph, where directed (binary) edges represent positive responses to survey questions about a specific sociometric relation. In this abstract setting, the mixed membership stochastic blockmodel (MMB) posits that a graph $G = (\mathcal{N}, Y)$ is drawn from the following procedure.

- For each node $p \in \mathcal{N}$:
  - Draw a K dimensional mixed membership vector $\vec{\pi}_p \sim$ Dirichlet ($\vec{\alpha}$).
- For each pair of nodes $(p,q) \in \mathcal{N} \times \mathcal{N}$:
  - Draw membership indicator for the initiator, $\vec{z}_{p\to q} \sim$ Multinomial ($\vec{\pi}_p$).
  - Draw membership indicator for the receiver, $\vec{z}_{q\to p} \sim$ Multinomial ($\vec{\pi}_q$).
  - Sample the value of their interaction, $Y(p,q) \sim$ Bernoulli ($\vec{z}_{p\to q}^{\top} B\, \vec{z}_{p\leftarrow q}$).

1. In previous work we combined mixed membership and blockmodels to perform analyses of a single collection of binary, paired measurements; namely, hypothesis testing, predicting and de-noising interactions within an unsupervised learning setting (Airoldi et al., 2005).
2. An indicator vector is used to denote membership in one of the K groups. Such a membership-indicator vector is specified as a K-dimensional vector of which only one element equals to one, whose index corresponds to the group to be indicated, and all other elements equal to zero.

This process is illustrated as a graphical model in Figure 1. Note that the group membership of each node is *context dependent*. That is, each node may assume different membership when interacting to or being interacted by different peers. Statistically, each node is an admixture of group-specific interactions. The two sets of latent group indicators are denoted by $\{\vec{z}_{p\to q} : p,q \in \mathcal{N}\} =: Z_{\to}$ and $\{\vec{z}_{p\leftarrow q} : p,q \in \mathcal{N}\} =: Z_{\leftarrow}$. Also note that the pairs of group memberships that underlie interactions need not be equal; this fact is useful for characterizing asymmetric interaction networks. Equality may be enforced when modeling symmetric interactions.

Under the MMB, the joint probability of the data Y and the latent variables $\{\vec{\pi}_{1:N}, Z_{\to}, Z_{\leftarrow}\}$ can be written in the following factored form,

$$p(Y,\vec{\pi}_{1:N},Z_{\to},Z_{\leftarrow}|\vec{\alpha},B) = \prod_{p,q} P(Y(p,q)|\vec{z}_{p\to q},\vec{z}_{p\leftarrow q},B)P(\vec{z}_{p\to q}|\vec{\pi}_p)P(\vec{z}_{p\leftarrow q}|\vec{\pi}_q)\prod_p P(\vec{\pi}_p|\vec{\alpha}). \quad (1)$$

This model generalizes to two important cases. First, multiple networks among the same actors can be generated by the same latent vectors. This may be useful, for instance, to analyze multivariate sociometric relations. Second, in the MMB the data generating distribution is a Bernoulli, but B can be a matrix that parameterizes any kind of distribution. This may be useful, for instance, to analyze collections of paired measurements, Y, that take values in an arbitrary metric space. We elaborate on this in Section 5.

2.1 Modeling Sparsity

Adjacency matrices encoding binary pairwise measurements are often sparse, that is, they contain many zeros or non-interactions. It is useful to distinguish two sources of non-interaction: they may be the result of the rarity of interactions in general, or they may be an indication that the pair of relevant blocks rarely interact. In applications to social sciences, for instance, nodes may represent people and blocks may represent social communities. It is reasonable to expect that a large portion of the non-interactions is due to limited opportunities of contact between people rather than due to deliberate choices, the structure of which the blockmodel is trying to estimate. It is useful to account for these two sources of sparsity at the model level. A good estimate of the portion of zeros that should not be explained by the blockmodel B reduces the bias of the estimates of its elements.

Thus, we introduce a sparsity parameter $\rho \in [0,1]$ in the MMB to characterize the source of non-interaction. Instead of sampling a relation $Y(p.q)$ directly the Bernoulli with parameter specified as above, we down-weight the probability of successful interaction to $(1-\rho)\cdot \vec{z}_{p\to q}^{\top} B\, \vec{z}_{p\leftarrow q}$. This is the result of assuming that the probability of a non-interaction comes from a mixture, $1-\sigma_{pq} = (1-\rho)\cdot \vec{z}_{p\to q}^{\top}(1-B)\, \vec{z}_{p\leftarrow q} + \rho$, where the weight ρ capture the portion zeros that should not be explained by the blockmodel B. A large value of ρ will cause the interactions in the matrix to be weighted more than non-interactions, in determining plausible values for $\{\vec{\alpha}, B, \vec{\pi}_{1:N}\}$.

The sparsity parameter ρ can be estimated. Its maximum likelihood estimate provides the best data-driven guess about the proportion of zeros that the blockmodel can explain. Introducing ρ provides a strategy to rescale B, by separating zeros in the adjacency matrix into those that are likely to be due to the blockmodel and those that are not.

2.2 Summarizing and De-Noising Pairwise Measurements

It is useful to distinguish two types of data analysis that can be performed with the mixed-membership blockmodel. First, MMB can be used to summarize the data, Y, in terms of the global blockmodel, B, and the node-specific mixed memberships, Πs. Second, MMB can be used to de-noise the data, Y, in terms of the global blockmodel, B, and interaction-specific single memberships, Zs. In both cases the model depends on a small set of unknown constants to be estimated: α, and B. The likelihood is the same in both cases, although, the rationale for including the set of latent variables Zs differs. When summarizing data, we could integrate out the Zs analytically; this leads to numerical optimization of a smaller set of variational parameters, Γs. We choose to keep the Zs to simplify inference. When de-noising, the Zs are instrumental in estimating posterior expectations of each interactions individually—a network analog to the Kalman Filter. The posterior expectations of an interaction is computed as follows, in the two cases,

$$\mathbb{E}\,[\,Y(p,q)=1\,] \approx \widehat{\vec{\pi}}_p{}' \widehat{B}\, \widehat{\vec{\pi}}_q \qquad \text{and} \qquad \mathbb{E}\,[\,Y(p,q)=1\,] \approx \widehat{\vec{\phi}}_{p\to q}{}' \widehat{B}\, \widehat{\vec{\phi}}_{p\leftarrow q}.$$

2.3 An Illustration: Crisis in a Cloister

To illustrate the MMB, we return to an analysis of the monk data described above. Sampson (1968) surveyed 18 novice monks in a monastery and asked them to rank the other novices in terms of four sociometric relations: like/dislike, esteem, personal influence, and alignment with the monastic credo. We consider Breiger's collation of Sampson's data (Breiger et al., 1975). The original graph of monk-monk interaction is illustrated in Figure 2 (left).

Sampson spent several months in a monastery in New England, where novice monks were preparing to join a monastic order. Sampson's original analysis was rooted in direct anthropological observations. He suggested the existence of tight factions among the novices: the loyal opposition (whose members joined the monastery first), the young turks (who joined later on), the outcasts (who were not accepted in the two main factions), and the waverers (who did not take sides). The events that took place during Sampson's stay at the monastery supported his observations—members of the young turks resigned or were expelled over religious differences (John and Gregory). We shall

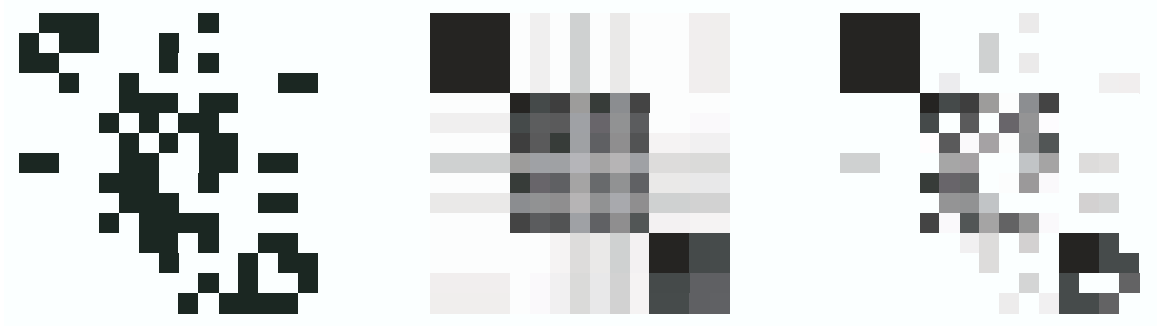


Figure 2: Original adjacency matrix of whom-do-like sociometric relations (left), relations predicted using approximate MLEs for $\vec{\pi}_{1:N}$ and B (center), and relations de-noised using the model including Zs indicators (right).

refer to the labels assigned by Sampson to the novices in the analysis below. For more analyses, we refer to Fienberg et al. (1985), Davis and Carley (2006) and Handcock et al. (2007).

Using the algorithms presented in Section 3, we fit the monks to MMB models for different numbers of groups, providing model estimates $\{\hat{\alpha}, \hat{B}\}$ and posterior mixed membership vectors $\vec{\pi}_n$ for each monk. Here, we use the following approximation to BIC to choose the number of groups in the MMB:

$$BIC = 2 \cdot \log p(Y) \approx 2 \cdot \log p(Y|\widehat{\vec{\pi}}, \widehat{Z}, \widehat{\vec{\alpha}}, \widehat{B}) - |\vec{\alpha}, B| \cdot \log |Y|,$$

which selects three groups, where $|\vec{\alpha}, B|$ is the number of hyper-parameters in the model, and $|Y|$ is the number of positive relations observed (Volinsky and Raftery, 2000; Handcock et al., 2007). Note that this is the same number of groups that Sampson identified. We illustrate the fit of model fit via the predicted network in Figure 2 (Right). The three panels contrast the different resolution of the original adjacency matrix of whom-do-like sociometric relations (left panel) obtained in different uses of MMB. If the goal of the analysis if to find a parsimonious summary of the data, the amount of relational information that is captured by in $\hat{\alpha}, \hat{B}$, and $\mathbb{E}[\vec{\pi}|Y]$ leads to a coarse reconstruction of the original sociomatrix (central panel). If the goal of the analysis if to de-noising a collection of pairwise measurements, the amount of relational information that is revealed by $\hat{\alpha}, \hat{B}, \mathbb{E}[\vec{\pi}|Y]$ and $\mathbb{E}[Z_{\rightarrow}, Z_{\leftarrow}|Y]$ leads to a finer reconstruction of the original sociomatrix, Y—relations in Y are re-weighted according to how much they *make sense* to the model (right panel).

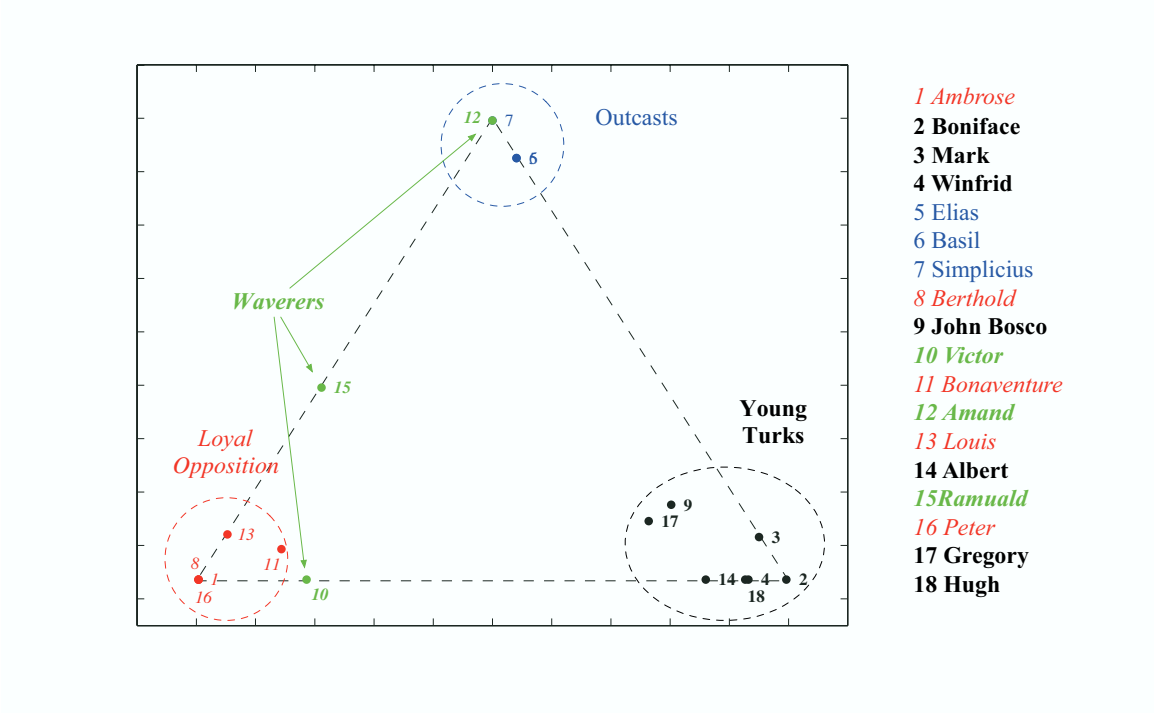


Figure 3: Posterior mixed membership vectors, $\vec{\pi}_{1:18}$, projected in the simplex. Numbered points can be mapped to monks' names using the legend on the right. The colors identify the four factions defined by Sampson's anthropological observations.

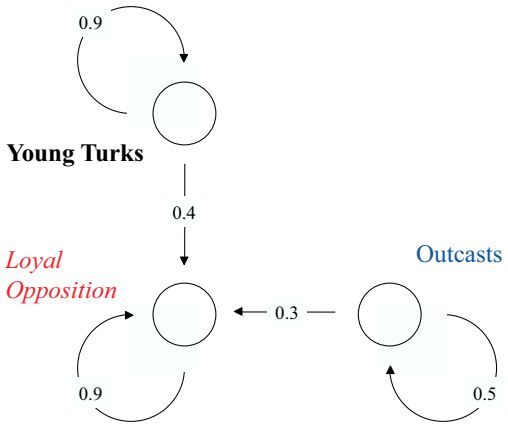


Figure 4: Estimated blockmodel in the monk data, $\hat{B}$.

The MMB provides interesting descriptive statistics about the actors in the observed graph. In Figure 3 we illustrate the the posterior means of the mixed membership scores, $\mathbb{E}[\vec{\pi}|Y]$, for the 18 monks in the monastery. Note that the monks cluster according to Sampson's classification, with Young Turks, Loyal Opposition, and Outcasts dominating each corner respectively. We can see the central role played by John Bosco and Gregory, who exhibit relations in all three groups, as well as the uncertain affiliations of Ramuald and Victor. (Amand's uncertain affiliation, however, is not captured.) The estimated blockmodel is shown in Figure 4.

3. Parameter Estimation and Posterior Inference

Two computational problems are central to the MMB: posterior inference of the per-node mixed membership vectors and per-pair roles, and parameter estimation of the Dirichlet parameters and Bernoulli rate matrix. We derive empirical Bayes estimates of the parameters $(\vec{\alpha}, B)$, and employ a mean-field approximation scheme for posterior inference.

3.1 Posterior Inference

The posterior inference problem is to compute the posterior distribution of the latent variables given a collection of observations. The normalizing constant of the posterior distribution is the marginal probability of the data, which requires an integral over the simplicial vectors $\vec{\pi}_p$,

$$p(Y|\vec{\alpha}, B) = \int_{\Pi} \sum_{Zs} \left(\prod_{p,q} P(Y(p,q)|\vec{z}_{p\to q}, \vec{z}_{p\leftarrow q}, B) P(\vec{z}_{p\to q}|\vec{\pi}_p) P(\vec{z}_{p\leftarrow q}|\vec{\pi}_q) \prod_{p} P(\vec{\pi}_p|\vec{\alpha}) \right) d\vec{\pi},$$

which is not solvable in closed form (Blei et al., 2003). A number of approximate inference algorithms for mixed membership models have appeared in recent years, including mean-field variational methods (Blei et al., 2003; Teh et al., 2007), expectation propagation (Minka and Lafferty, 2002), and Monte Carlo Markov chain sampling (MCMC) (Erosheva and Fienberg, 2005; Griffiths and Steyvers, 2004).

We appeal to variational methods (Jordan et al., 1999; Wainwright and Jordan, 2003). The main idea behind variational methods is to first posit a distribution of the latent variables with free parameters, and then fit those parameters such that the distribution is close in Kullback-Leibler divergence to the true posterior. The variational distribution is simpler than the true posterior so that the optimization problem can be approximately solved. Good reviews of variational methods can be found in Wainwright and Jordan (2003), Xing et al. (2003), Bishop et al. (2003) and Airoldi (2007).

In the MMB, we begin by bounding the log of the marginal probability of the data with Jensen's inequality,

$$\log p(Y \mid \alpha, B) \geq \mathbb{E}_q \left[\log p(Y, \vec{\pi}_{1:N}, Z_{\rightarrow}, Z_{\leftarrow} | \alpha, B) \right] - \mathbb{E}_q \left[\log q(\vec{\pi}_{1:N}, Z_{\rightarrow}, Z_{\leftarrow}) \right].$$

We have introduced a distribution of the latent variables q that depends on a set of free parameters. We specify q as the mean-field fully-factorized family,

$$q(\vec{\pi}_{1:N}, Z_{\rightarrow}, Z_{\leftarrow} | \vec{\gamma}_{1:N}, \Phi_{\rightarrow}, \Phi_{\leftarrow}) = \prod_p q_1(\vec{\pi}_p | \vec{\gamma}_p) \prod_{p,q} \left(q_2(\vec{z}_{p \rightarrow q} | \vec{\phi}_{p \rightarrow q}) \, q_2(\vec{z}_{p \leftarrow q} | \vec{\phi}_{p \leftarrow q}) \right),$$

where q_1 is a Dirichlet, q_2 is a multinomial, and $\{\vec{\gamma}_{1:N}, \Phi_{\rightarrow}, \Phi_{\leftarrow}\}$ are the set of free *variational parameters* that are optimized to tighten the bound.

Tightening the bound with respect to the variational parameters is equivalent to minimizing the KL divergence between q and the true posterior. When all the nodes in the graphical model are conjugate pairs or mixtures of conjugate pairs, we can directly write down a coordinate ascent algorithm for this optimization to reach a local maximum of the bound. The updates for the variational multinomial parameters are

$$\hat{\phi}_{p \rightarrow q, g} \propto e^{\mathbb{E}_q[\log \pi_{p,g}]} \cdot \prod_h \left(B(g,h)^{Y(p,q)} \cdot (1 - B(g,h))^{1 - Y(p,q)} \right)^{\phi_{p \leftarrow q, h}} \tag{2}$$

$$\hat{\phi}_{p \leftarrow q, h} \propto e^{\mathbb{E}_q[\log \pi_{q,h}]} \cdot \prod_g \left(B(g,h)^{Y(p,q)} \cdot (1 - B(g,h))^{1 - Y(p,q)} \right)^{\phi_{p \rightarrow q, g}}, \tag{3}$$

for $g, h = 1, \ldots, K$. The update for the variational Dirichlet parameters $\gamma_{p,k}$ is

$$\hat{\gamma}_{p,k} = \alpha_k + \sum_q \phi_{p \rightarrow q, k} + \sum_q \phi_{p \leftarrow q, k}, \tag{4}$$

for all nodes $p = 1, \ldots, N$ and $k = 1, \ldots, K$. The complete coordinate ascent algorithm is described in Figure 5.

To improve convergence, we employed a nested variational inference scheme based on an alternative schedule of updates to the traditional ordering. In a typical schedule for coordinate ascent (which we call "naïve variational inference"), one initializes the variational Dirichlet parameters $\vec{\gamma}_{1:N}$ and the variational multinomial parameters $(\vec{\phi}_{p \rightarrow q}, \vec{\phi}_{p \leftarrow q})$ to non-informative values, and then iterates the following two steps until convergence: (i) update $\vec{\phi}_{p \rightarrow q}$ and $\phi_{p \leftarrow q}$ for all edges (p,q), and (ii) update $\vec{\gamma}_p$ for all nodes $p \in \mathcal{N}$. In such algorithm, at each variational inference cycle we need to allocate $NK + 2N^2K$ scalars.

In our experiments, the naïve variational algorithm often converged only after many iterations. We attribute this behavior to the dependence between $\vec{\gamma}_{1:N}$ and B, which is not satisfied by the naïve algorithm. Some intuition about why this may happen follows. From a purely algorithmic

1. initialize $\vec{\gamma}^0_{pk} = \frac{2N}{K}$ for all p,k
2. **repeat**
3. **for** $p = 1$ to N
4. **for** $q = 1$ to N
5. get **variational** $\vec{\phi}^{t+1}_{p\to q}$ and $\vec{\phi}^{t+1}_{p\leftarrow q} = f\ (\ Y(p,q), \vec{\gamma}^t_p, \vec{\gamma}^t_q, B^t\ )$
6. partially update $\vec{\gamma}^{t+1}_p$, $\vec{\gamma}^{t+1}_q$ and B^{t+1}
7. **until** convergence

5.1. initialize $\phi^0_{p\to q,g} = \phi^0_{p\leftarrow q,h} = \frac{1}{K}$ for all g,h
5.2. **repeat**
5.3. **for** $g = 1$ to K
5.4. update $\phi^{s+1}_{p\to q} \propto f_1\ (\ \vec{\phi}^s_{p\leftarrow q}, \vec{\gamma}_p, B\ )$
5.5. normalize $\vec{\phi}^{s+1}_{p\to q}$ to sum to 1
5.6. **for** $h = 1$ to K
5.7. update $\phi^{s+1}_{p\leftarrow q} \propto f_2\ (\ \vec{\phi}^s_{p\to q}, \vec{\gamma}_q, B\ )$
5.8. normalize $\vec{\phi}^{s+1}_{p\leftarrow q}$ to sum to 1
5.9. **until** convergence

Figure 5: **Top:** The two-layered variational inference for $(\vec{\gamma}, \phi_{p\to q,g}, \phi_{p\leftarrow q,h})$ and $M = 1$. The inner algorithm consists of Step 5. The function f is described in details in the bottom panel. The partial updates in Step 6 for $\vec{\gamma}$ and B refer to Equation 4 of Section B.4 and Equation 5 of Section B.5, respectively. **Bottom:** Inference for the variational parameters $(\vec{\phi}_{p\to q}, \vec{\phi}_{p\leftarrow q})$ corresponding to the basic observation $Y(p,q)$. This nested algorithm details Step 5 in the top panel. The functions f_1 and f_2 are the updates for $\phi_{p\to q,g}$ and $\phi_{p\leftarrow q,h}$ described in Equations 2 and 3 of Section B.4.

perspective, the naïve variational EM algorithm instantiates a large coordinate ascent algorithm, where the parameters can be divided into blocks. Blocks are processed in a specific order, and the parameters within each block get all updated each time.[3] At every new iteration the naïve algorithm sets all the elements of $\vec{\gamma}^{t+1}_{1:N}$ equal to the same constant. This dampens the likelihood by suddenly breaking the dependence between the estimates of parameters in $\hat{\vec{\gamma}}^t_{1:N}$ and in $\hat{B}^t$ that was being inferred from the data during the previous iteration.

Instead, the nested variational inference algorithm maintains some of this dependence that is being inferred from the data across the various iterations. This is achieved mainly through a different

3. Within a block, the order according to which (scalar) parameters get updated is not expected to affect convergence.

scheduling of the parameter updates in the various blocks. To a minor extent, the dependence is maintained by always keeping the block of free parameters, $(\vec{\phi}_{p\to q}, \vec{\phi}_{p\leftarrow q})$, optimized given the other variational parameters. Note that these parameters are involved in the updates of parameters in $\vec{\gamma}_{1:N}$ and in B, thus providing us with a channel to maintain some of the dependence among them, that is, by keeping them at their optimal value given the data.

Furthermore, the nested algorithm has the advantage that it trades time for space thus allowing us to deal with large graphs; at each variational cycle we need to allocate $NK+2K$ scalars only. The increased running time is partially offset by the fact that the algorithm can be parallelized and leads to empirically observed faster convergence rates.

An alternative strategy to perform inference is given by Monte Carlo Markov chain (e.g., see Griffiths and Steyvers, 2004; Kemp et al., 2004). While powerful in some settings, MCMC is impractical here. There are too many variables to sample. The proposed nested variational EM algorithm outperforms MCMC variations (i.e., blocked and collapsed Gibbs samplers) in terms of memory requirements and convergence rates.

3.2 Parameter Estimation

We compute the empirical Bayes estimates of the model hyper-parameters $\{\vec{\alpha}, B\}$ with a variational expectation-maximization (EM) algorithm. Alternatives to empirical Bayes have been proposed to fix the hyper-parameters and reduce the computation. The results, however, are not always satisfactory and often times cause of concern, since the inference is sensitive to the choice of the hyper-parameters (Joutard et al., 2007). Empirical Bayes, on the other hand, guides the posterior inference towards a region of the hyper-parameter space that is supported by the data.

Variational EM uses the lower bound in Equation 5 as a surrogate for the likelihood. To find a local optimum of the bound, we iterate between fitting the variational distribution q to approximate the posterior and maximizing the corresponding bound with respect to the parameters. The latter M-step is equivalent to finding the MLE using expected sufficient statistics under the variational distribution. We consider the maximization step for each parameter in turn.

A closed form solution for the approximate maximum likelihood estimate of $\vec{\alpha}$ does not exist (Minka, 2003). We use a linear-time Newton-Raphson method, where the gradient and Hessian are

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\vec{\alpha}}}{\partial \alpha_k} &= N\left(\psi\left(\sum_k \alpha_k\right) - \psi(\alpha_k)\right) + \sum_p \left(\psi(\gamma_{p,k}) - \psi\left(\sum_k \gamma_{p,k}\right)\right), \\
\frac{\partial \mathcal{L}_{\vec{\alpha}}}{\partial \alpha_{k_1}\alpha_{k_2}} &= N\left(\mathbb{I}_{(k_1=k_2)} \cdot \psi'(\alpha_{k_1}) - \psi'\left(\sum_k \alpha_k\right)\right).
\end{aligned}$$

The approximate MLE of B is

$$\hat{B}(g,h) = \frac{\sum_{p,q} Y(p,q) \cdot \phi_{p\to qg}\, \phi_{p\leftarrow qh}}{(1-\rho) \cdot \sum_{p,q} \phi_{p\to qg}\, \phi_{p\leftarrow qh}},$$

for every index pair $(g,h) \in [1,K] \times [1,K]$. Finally, the approximate MLE of the sparsity parameter ρ is

$$\hat{\rho} = \frac{\sum_{p,q} \left(1 - Y(p,q)\right) \cdot \left(\sum_{g,h} \phi_{p\to qg}\, \phi_{p\leftarrow qh}\right)}{\sum_{p,q} \sum_{g,h} \phi_{p\to qg}\, \phi_{p\leftarrow qh}}.$$

Alternatively, we can fix ρ prior to the analysis; the density of the interaction matrix is estimated with $\hat{d} = \sum_{p,q} Y(p,q)/N^2$, and the sparsity parameter is set to $\tilde{\rho} = (1 - \hat{d})$. This latter estimator attributes all the information in the non-interactions to the point mass, that is, to latent sources other than the block model B or the mixed membership vectors $\vec{\pi}_{1:N}$. It does however provide a quick recipe to reduce the computational burden during exploratory analyses.[4]

Several model selection strategies are available for complex hierarchical models (Joutard et al., 2007). In our setting, model selection translates into the determination of a plausible value of the number of groups K. In the various analyses presented, we selected the optimal value of K according to two strategies. On large networks, we selected K corresponding to the highest averaged held-out likelihood in a cross-validation experiment. On small networks—where cross-validation cannot be expected to work well, as we discuss in Section 5—we selected K using an approximation to BIC.

4. Experiments and Results

We present a study of simulated data and applications to social and protein interaction networks.

Simulations are performed in Section 4.1 to show that both mixed membership, $\vec{\pi}_{1:N}$, and the latent block structure, B, can be recovered from data, when they exist, and that the nested variational inference algorithm is faster than the naïve implementation while reaching the same peak in the likelihood—all other things being equal.

The application to a friendship network among students in Section 4.2 tests the model on a real data set where we expect a well-defined latent block structure to inform the observed connectivity patterns in the network. In this application, the blocks are interpretable in terms of grades. We compare our results with those that were recently obtained with a simple mixture of blocks (Doreian et al., 2007) and with a latent space model (Handcock et al., 2007) on the same data.

The application to a protein interaction network in Section 4.3 tests the model on a real data set where we expect a noisy, vague latent block structure to inform the observed connectivity patterns in the network to some degree. In this application, the blocks are interpretable in terms functional biological contexts. This application tests to what extent our model can reduce the dimensionality of the data, while revealing substantive information about the functionality of proteins that can be used to inform subsequent analyses.

4.1 Exploring Expected Model Behavior with Simulations

In developing the MMB and the corresponding computation, our hope is the the model can recover both the mixed membership of nodes to clusters and the latent block structure among clusters in situations where a block structure exists and the relations are measured with some error. To substantiate this claim, we sampled graphs of $100, 300,$ and 600 nodes from blockmodels with $4, 10,$ and 20 clusters, respectively, using the MMB. We used different values of α to simulate a range of settings in terms of membership of nodes to clusters—from unique $(\alpha = 0.05)$ to mixed $(\alpha = 0.25)$.

Recovering the truth. The variational EM algorithm successfully recovers both the latent block model B and the latent mixed membership vectors $\vec{\pi}_{1:N}$. In Figure 6 we show the adjacency matrices of binary interactions where rows, that is, nodes, are reordered according to their most likely membership. The estimated reordering reveals the block model that was originally used to simulate

4. Note that $\tilde{\rho} = \hat{\rho}$ in the case of single membership. In fact, that implies $\phi^m_{p \to qg} = \phi^m_{p \leftarrow qh} = 1$ for some (g,h) pair, for any (p,q) pair.

the interactions. As α increases, each node is likely to belong to more clusters. As a consequence, they express interaction patterns of clusters. This phenomenon reflects in the reordered interaction matrices as the block structure is less evident.

Nested variational inference. The nested variational algorithm drives the log-likelihood to converge faster to its peak than the naïve algorithm. In Figure 7 (left panel) we compare the running times of the nested variational-EM algorithm versus the naïve implementation. The nested algorithm, which is more efficient in terms of space, converged faster. Furthermore, the nested variational algorithm can be parallelized given that the updates for each interaction (i, j) are independent of one another.

Choosing the number of blocks. The right panel of Figure 7 shows an example where cross-validation is sufficient to perform model selection for the MMB. The example shown corresponds to a network among 300 nodes with $K = 10$ clusters. We measure the number of latent clusters

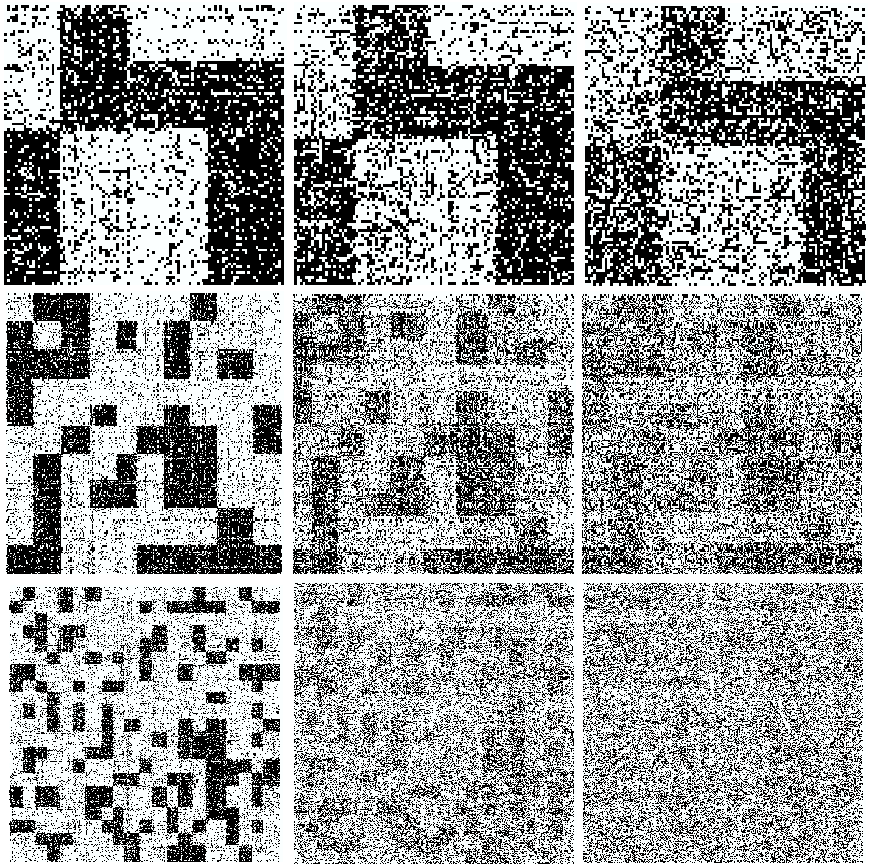


Figure 6: Adjacency matrices of corresponding to simulated interaction graphs with 100 nodes and 4 clusters, 300 nodes and 10 clusters, 600 nodes and 20 clusters (top to bottom) and α equal to $0.05, 0.1$ and 0.25 (left to right). Rows, which corresponds to nodes, are reordered according to their most likely membership. The estimated reordering accurately reveals the original blockmodel.

on the X axis and the average held-out log-likelihood, corresponding to five-fold cross-validation experiments, on the Y axis. The nested variational EM algorithm was xrun until convergence, for each value of K we tested, with a tolerance of $\varepsilon = 10^{-5}$. Our estimate for K occurs at the peak in the average held-out log-likelihood, and equals the correct number of clusters, $K^* = 10$

4.2 Application to Social Network Analysis

We considered a friendship network among a group of 69 students in grades 7–12. The analysis here directly compares clustering results obtained by MMB to published clustering results obtained by competing models, in a setting where a fair amount of social segregation is expected (Doreian et al., 2007; Handcock et al., 2007).

The National Longitudinal Study of Adolescent Health is nationally representative study that explores the how social contexts such as families, friends, peers, schools, neighborhoods, and communities influence health and risk behaviors of adolescents, and their outcomes in young adulthood (Harris et al., 2003; Udry, 2003). As part of the survey, a questionnaire was administered to a sample of students in each school, who were allowed to nominate up to 10 friends. We analyzed a friendship network among the students, at the same school that was considered by Handcock et al. (2007) and discussants. Friendship nominations were collected among 71 students in grades 7 to 12; two students did not nominate any friends. The network of binary, asymmetric friendship relations among the remaining 69 students that constitutes our data is shown in Figure 9 (left).

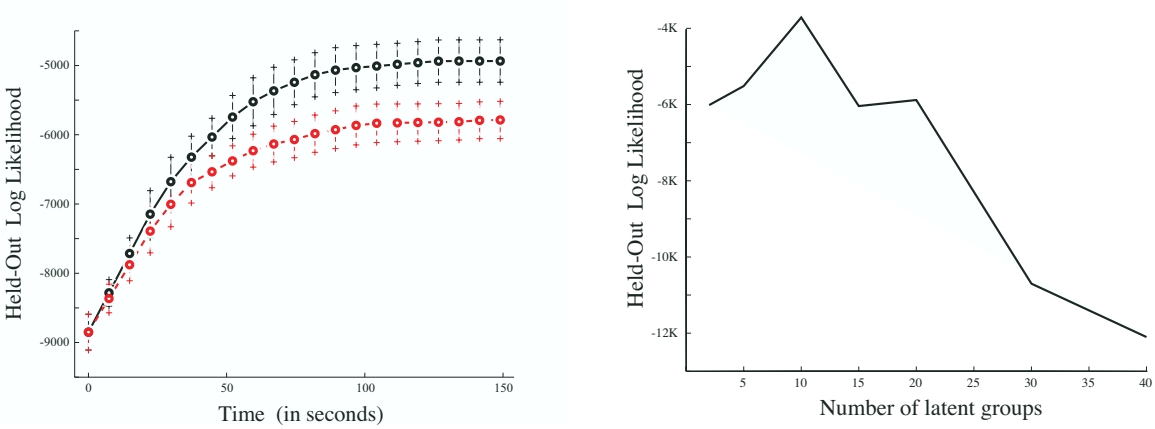


Figure 7: **Left:** The running time of the naïve variational inference (dashed, red line) against the running time of our enhanced (nested) variational inference algorithm (solid, black line), on a graph with 100 nodes and 4 clusters. We measure the number of seconds on the X axis and the log-likelihood on the Y axis. The two curves are averages over 26 experiments, and the error bars are at three standard deviations. Each of the 26 pairs of experiments was initialized with the same values for the parameters. **Right:** The held-out log-likelihood is indicative of the true number of latent clusters, on simulated data. We measure the number of latent clusters on the X axis and the log-likelihood on a test set on the Y axis. In the example shown, the peak identifies the correct number of clusters, $K^* = 10$

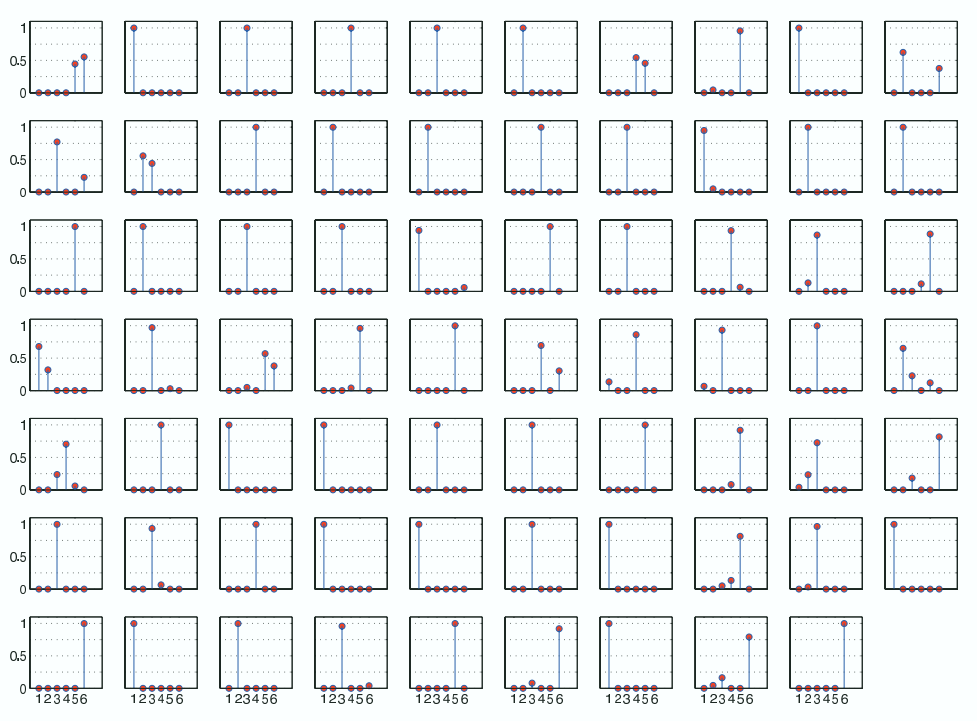


Figure 8: The posterior mixed membership scores, $\vec{\pi}$, for the 69 students. Each panel correspond to a student; we order the clusters 1 to 6 on the X axis, and we measure the student's grade of membership to these clusters on the Y axis.

Given the size of the network we used BIC to perform model selection, as in the monks example of Section 2.3. The results suggest a model with $K^* = 6$ groups. (We fix $K^* = 6$ in the analyses that follow.) The hyper-parameters estimated with the nested variational EM. They are $\hat{\alpha} = 0.0487$, $\hat{\rho} = 0.936$, and a fairly diagonal blockmodel,

$$\hat{B} = \begin{bmatrix} 0.3235 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.3614 & 0.0002 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.2607 & 0.0 & 0.0 & 0.0002 \\ 0.0 & 0.0 & 0.0 & 0.3751 & 0.0009 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0002 & 0.3795 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.3719 \end{bmatrix}.$$

Figure 8 shows the expected posterior mixed membership scores for the 69 students in the sample; few students display mixed membership. The rarity of mixed membership in this context is expected, while mixed membership may signal unexpected social situations for further investigation. For instance, it may signal a family bond such as brotherhood, or a student that is repeating a grade and is thus part of a broader social clique. In Figure 9, we contrast the friendship relation data (left) to the estimates obtained by thresholding the estimated probabilities of a relation, using the blockmodel and the node-specific latent variables (center) and the interactions-specific latent variables (right). The model provides a good summary of the social structure in the school; students

tend to befriend other students in the same grade, with a few exceptions. The low degree of mixed membership explains the absence of obvious differences between the model-based reconstructions of the friendship relations with the two model variants (center and right).

Figure 9: Original matrix of friensdhip relations among 69 students in grades 7 to 12 (left), and friendship estimated relations obtained by thresholding the posterior expectations $\vec{\pi}_p{}' B \vec{\pi}_q | Y$ (center), and $\vec{\phi}_p{}' B \vec{\phi}_q | Y$ (right).

Next, we attempted a quantitative evaluation of the goodness of fit. In this data, the blocks are clearly interpretable a-posteriori in terms of grades. The mixed membership vectors provide a mapping between grades and blocks. Conditionally on such a mapping, we assign students to the grade they are most associated with, according to their posterior-mean mixed membership vectors, $\mathbb{E}[\vec{\pi}_n | Y]$. To be fair in the comparison with competing models, we assign students to a unique grade—despite MMB allows for mixed membership. Table 1 computes the correspondence of grades to blocks by quoting the number of students in each grade-block pair, for MMB versus the mixture blockmodel (MB) in Doreian et al. (2007), and the latent space cluster model (LSCM) in Handcock et al. (2007). The higher the sum of counts on diagonal elements is the better is the correspondence, while the higher the sum of counts off diagonal elements is the worse is the correspondence. MMB performs best by allocating 63 students to their grades, versus 57 of MB, and 37 of LSCM. Correspondence only partially captures goodness of fit, however, it is a good metric in the setting we consider, where a fair amount of clustering is present. The results suggest that the extra-flexibility MMB offers over MB and LSCM reduces bias in the prediction of the membership of students to blocks. In other words, mixed membership does not absorb noise in this example; rather it accommodates variability in the friendship relation that is instrumental in producing better predictions.

Concluding this example, we note how the model decouples the observed friendship patterns into two complementary sources of variability. On the one hand, the connectivity matrix B is a global, unconstrained set of hyper-parameters. On the other hand, the mixed membership vectors $\vec{\pi}_{1:N}$ provide a collection of node-specific latent vectors, which inform the directed connections in the graph in a symmetric fashion.

4.3 Application to Protein Interactions in *Saccharomyces Cerevisiae*

We considered physical interactions among 871 proteins in yeast. The analysis allows us to evaluate the utility of MMB in summarizing and de-noising complex connectivity patterns quantitatively, using an independent set of functional annotations. For instance, between two models that sug-

	MMB Clusters						MSB Clusters						LSCM Clusters					
Grade	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
7	13	1	0	0	0	0	13	1	0	0	0	0	13	1	0	0	0	0
8	0	9	2	0	0	1	0	10	2	0	0	0	0	11	1	0	0	0
9	0	0	16	0	0	0	0	0	10	0	0	6	0	0	7	6	3	0
10	0	0	0	10	0	0	0	0	0	10	0	0	0	0	0	0	3	7
11	0	0	1	0	11	1	0	0	1	0	11	1	0	0	0	0	3	10
12	0	0	0	0	0	4	0	0	0	0	0	4	0	0	0	0	0	4

Table 1: Grade levels versus (highest) expected posterior membership for the 69 students, according to three alternative models. MMB is the proposed mixed membership stochastic blockmodel, MSB is a simpler stochastic block mixture model (Doreian et al., 2007), and LSCM is the latent space cluster model (Handcock et al., 2007).

gest different sets of interactions as reliable, we prefer the model that reveals *functionally relevant* interactions—as measured using the annotations.

Protein interactions (PPI) form the physical basis for the formation of stable protein complexes (i.e., protein clusters) and signaling pathways (i.e., cascades of protein interaction events) that carry out all major biological processes in the cell. A number of high-throughput experimental technologies have been devised to determine the set of interacting proteins on a global scale in yeast. These include two-hybrid (Y2H) screens and mass spectrometry methods (Gavin et al., 2002; Ho et al., 2002; Krogan et al., 2006). High-throughput technologies, however, often miss to identify interactions that are not present under the given conditions. Specific wet-lab methods employed by a certain technology, such as tagging, may disturb the formation of a stable protein complex, and weakly associated components may dissociate and escape detection. Statistical models that encode information about functional processes with high precision are an essential tool for carrying out probabilistic de-noising of biological signals from high-throughput experiments.

The goal of the analysis of protein interactions with MMB is to reveal the proteins' diverse functional roles by analyzing their local and global patterns of interaction. The biochemical composition of individual proteins make them suitable for carrying out a specific set of cellular operations, or *functions*. The main intuition behind our methodology is that pairs of protein interact because they participate in the same cellular process, as part of the same stable protein complex, that is, co-location, or because they are part of interacting protein complexes, as they carry out compatible cellular operations (Alberts et al., 2002). Below, we describe the MIPS protein interactions data and the possible interpretations of the blocks in MMB in terms of biological functions, and we report results of two experiments.

4.3.1 PROTEIN INTERACTION DATA AND FUNCTIONAL ANNOTATION DATA

The Munich Institute for Protein Sequencing (MIPS) database was created in 1998 based on evidence derived from a variety of experimental techniques (Mewes et al., 2004). It includes a hand-curated collection of protein interactions that does not include interactions obtained with high-throughput technologies. The collection covers about 8000 protein complex associations in yeast.

We analyzed a subset of this collection containing 871 proteins, the interactions amongst which were hand-curated.

The MIPS institute also provides a set of functional annotations for each protein. These annotations are organized in a tree, with 15 nodes (i.e., high-level functions) at the first level, 72 nodes (i.e., the mid-level functions) at the second level, and 255 nodes (i.e., the low-level functions) at the the leaf level. We mapped the 871 proteins in our collections to the high-level functions of the MIPS annotation tree. Table 2 quotes the number of proteins annotated to each of these 15 functions. Most proteins participate in more than one functional category, with an average of ≈ 2.4 functional annotations for each protein.. The relative importance of functional categories in our collection, in terms of the number of proteins involved, is similar to the relative importance of functional categories over the entire MIPS collection. We can also represent each protein in terms of its MIPS functional annotations. This leads to a 15-dimensional, binary representation for each protein, $\vec{b}_p$, where a component $\vec{b}_p(k) = 1$ indicates that protein p is annotated with function k in Table 2. Figure 10 shows the binary representations, $\vec{b}_{1:871}$, of the proteins in our collections; each panel corresponds to a protein; the 15 functional categories are ordered as in Table 2 on the X axis, whereas the presence or absence of the corresponding functional annotation is displayed on the Y axis. In Section 4.3.2, we fit a mixed membership blockmodel with $K = 15$, and we explore the direct correspondence between protein-specific mixed memberships to blocks, $\vec{\pi}_{1:871}$, and MIPS-derived functional annotations, $\vec{b}_{1:871}$.

An alternative source of functional annotations is the gene ontology (GO), distributed as part of the Saccharomyces genome database (Ashburner et al., 2000). GO provides vocabularies for describing the molecular function, biological process, and cellular component of gene products—such as proteins. Terms are organized in a directed acyclic graph. Terms at the top represent broader, more general concepts, terms lower down represent more specific concepts. There are two different relationship types between (parent-child) terms: "is a" and "part of". Proteins are annotated to terms, and, most importantly, a protein is typically annotated to multiple terms, in different portions of the GO annotation graph. We restrict our evaluations to a collection of GO terms that is specific enough for a co-annotation (i.e., two proteins annotated to the same term) to be functionally relevant to molecular biologists (Myers et al., 2006). In Section 4.3.3, we select the mixed membership blockmodel best for predicting out-of-sample interactions, corresponding to

#	Category	Count	#	Category	Count
1	Metabolism	125	9	Interaction w/ cell. environment	18
2	Energy	56	10	Cellular regulation	37
3	Cell cycle & DNA processing	162	11	Cellular other	78
4	Transcription (tRNA)	258	12	Control of cell organization	36
5	Protein synthesis	220	13	Sub-cellular activities	789
6	Protein fate	170	14	Protein regulators	1
7	Cellular transportation	122	15	Transport facilitation	41
8	Cell rescue, defence & virulence	6			

Table 2: The 15 high-level functional categories obtained by cutting the MIPS annotation tree at the first level and how many proteins (out of 871) participate in each.

$K^* = 50$, and we explore its goodness-of-fit indirectly—rather than attempting a direct interpretation of the model's parameters—, in terms of the number of predicted interactions that are functionally relevant according to GO functional annotations.

4.3.2 DIRECT EVALUATION: THE MODEL CAPTURES SUBSTANTIVE BIOLOGY

In the first experiment, we fit a model with $K = 15$ blocks, and we attempt a direct interpretation of the blocks in terms of the 15 high-level functional categories in the MIPS annotation tree—separate from the MIPS protein interaction data, and independently conceived. We discuss results

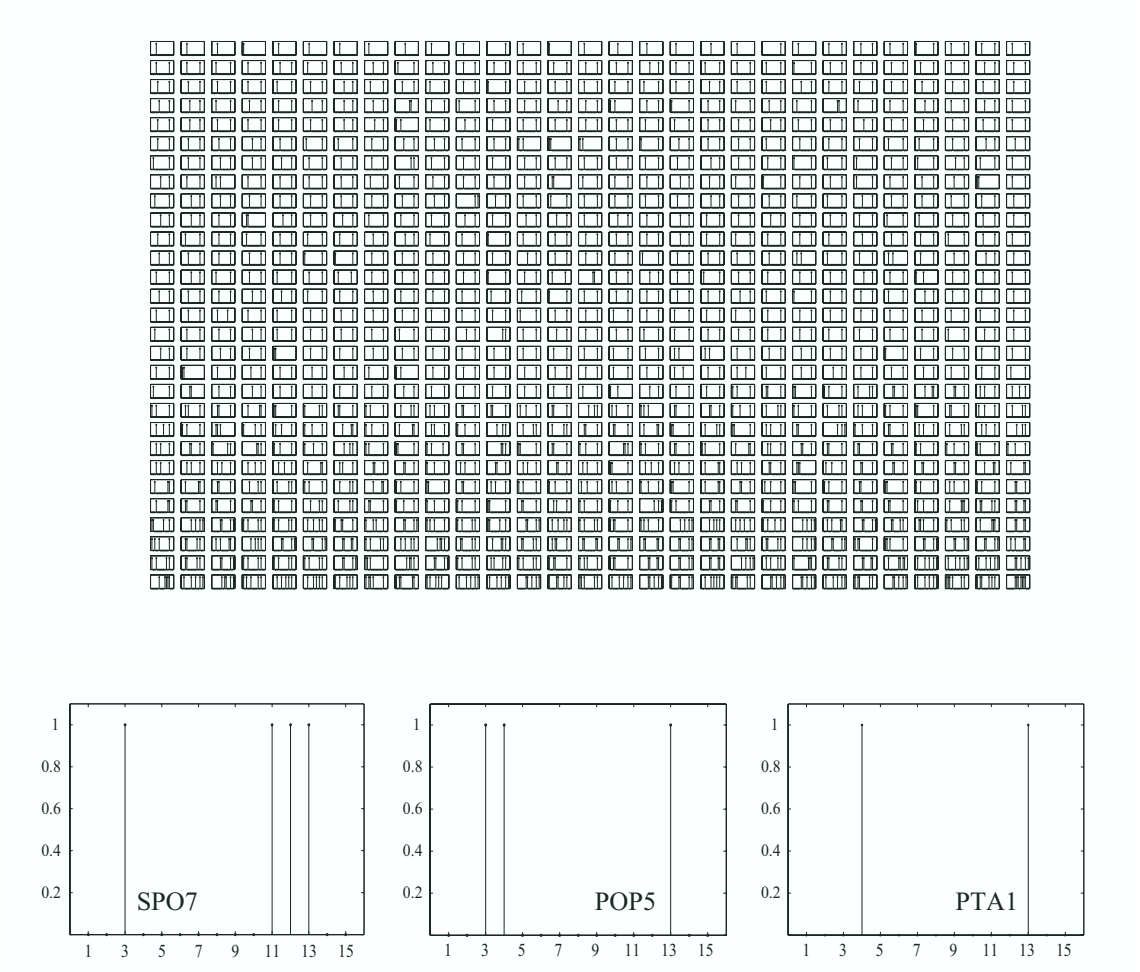


Figure 10: By mapping individual proteins to the 15 general functions in Table 2, we obtain a 15-dimensional representation for each protein. Here, each panel corresponds to a protein; the 15 functional categories are displayed on the X axis, whereas the presence or absence of the corresponding functional annotation is displayed on the Y axis. The plots at the bottom zoom into three example panels (proteins).

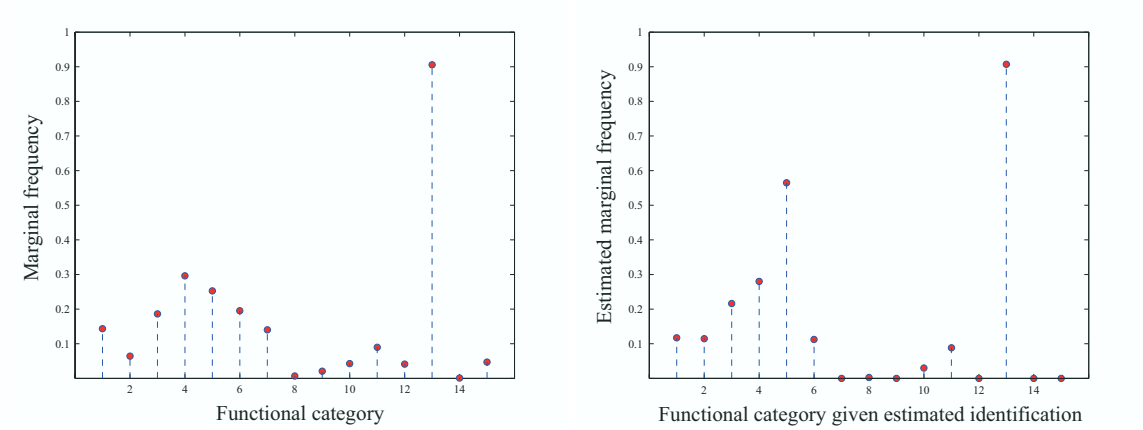


Figure 11: The mapping of blocks to functions is estimated by maximizing the accuracy of the predicted annotations of 87 proteins. We plot marginal frequencies of proteins' membership to true functions (left) and to predicted functions (right).

that portray the relevance of mixed membership, the resolution of the identification of blocks with functional categories, and selected predictions.

We want to compute the correspondence between protein-specific mixed memberships to blocks, $\vec{\pi}_{1:871}$, and MIPS-derived functional annotations, $\vec{b}_{1:871}$. The $K = 15$ blocks in the blockmodel B are not directly identifiable in terms of functional categories. In other words, we need to estimate a permutation of the components of $\vec{\pi}_n$ in order to be able to interpret $E[\pi_n(k)|Y]$ as the expected degree of membership of protein n in function k of Table 2—rather than simply the expected degree of membership of protein n in block k, out of 15. To estimate the permutation that best identifies blocks to functions, we proceeded as follows. We sampled 87 proteins and their corresponding MIPS annotations, $\vec{b}_{1:87}$. We predicted membership of the 87 proteins by thresholding their mixed membership representations,

$$\hat{b}_n(k) = \begin{cases} 1 & \text{if } \pi_n(k) > \tau \\ 0 & \text{otherwise,} \end{cases}$$

where τ is the 95th percentile of the ensemble of elements of $\vec{\pi}_{1:87}$, corresponding to the 87 proteins in the training set. We then greedily identified the mapping that maximizing the accuracy of the predicted annotations of 87 proteins. We used this mapping to compare predicted versus known functional annotations for all proteins; in Figure 11 we plot marginal frequencies of proteins' membership to true functions (left panel) and to predicted functions (right panel). The accuracy on the 90% testing set is about 87%. An algorithm that randomly guesses annotations, knowing the right proportions of annotations in each category, leads to a baseline accuracy of about 70%. Figure 12 shows predicted mixed memberships (dashed, red lines) versus the true annotations (solid, black lines), given the estimated mapping of blocks to functions, for six example proteins.

4.3.3 INDIRECT EVALUATION: FUNCTIONAL CONTENT OF PREDICTED INTERACTIONS

In the second experiment, we selected the mixed membership blockmodel best for predicting out-of-sample interactions, and we explored its goodness-of-fit indirectly, in terms of the number of

predicted interactions that are functionally relevant according to GO present in estimated protein interaction networks obtained with the two types of analyses that MMB supports; summarization and de-noising.

We fit models with K ranging between 2 and 255. We selected the best model ($K = 50$) using cross-validated held-out log likelihood, as in Figure 7. This finding supports the hypothesis that proteins derived from the MIPS data are interpretable in terms functional biological contexts. Alternatively, the blocks might encode signal at a finer resolution, such as that of protein complexes.

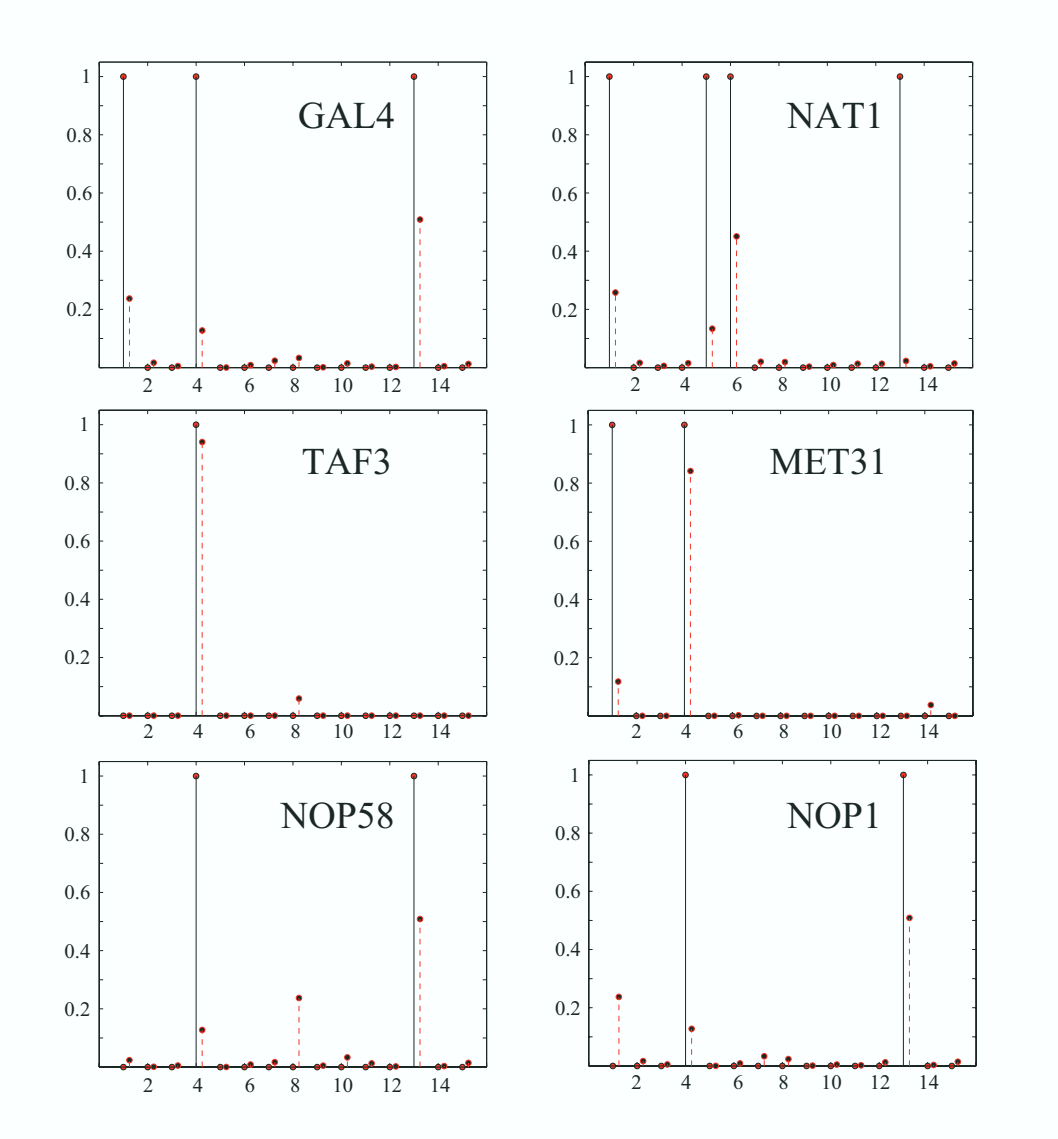


Figure 12: Predicted mixed-memberships (dashed, red lines) versus binary manually curated functional annotations (solid, black lines) for six example proteins, given the estimated mapping of blocks to functions in Figure 11.

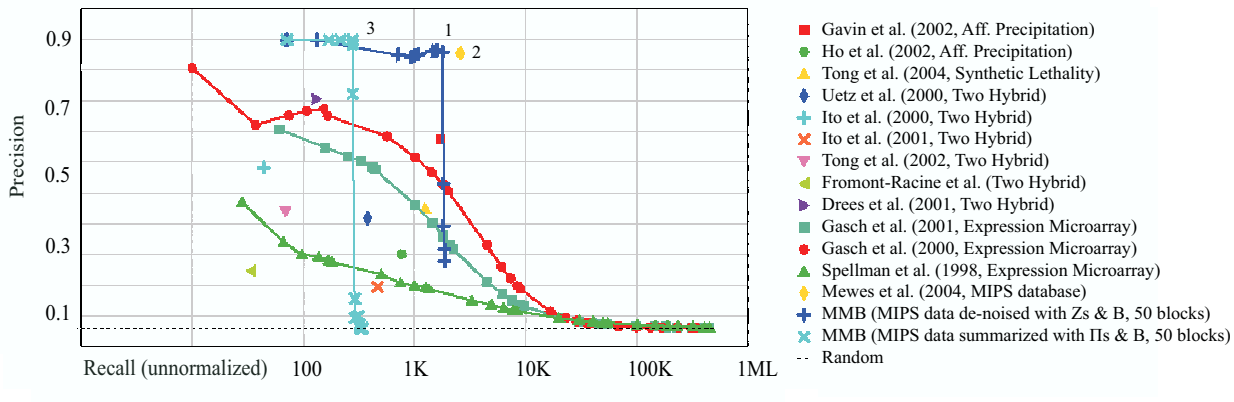


Figure 13: In the top panel we measure the functional content of the the MIPS collection of protein interactions (yellow diamond), and compare it against other published collections of interactions and microarray data, and to the posterior estimates of the MMB models—computed as described in Section 4.3.3. A breakdown of three estimated interaction networks (the points annotated 1, 2, and 3) into most represented gene ontology categories is detailed in Table 3.

If that was the case, however, we would expect the optimal number of blocks to be significantly higher; $871/5 \approx 175$, given an average size of five proteins in a complex (Krogan et al., 2006).

Using this model, we computed posterior model-based expectations of each interaction as follows,

$$\mathbb{E}\,[\,Y(p,q)\,] \approx \widehat{\vec{\pi}}_p{}' \widehat{B}\, \widehat{\vec{\pi}}_q \qquad \text{and} \qquad \mathbb{E}\,[\,Y(p,q)\,] \approx \widehat{\vec{\phi}}_{p\to q}{}' \widehat{B}\, \widehat{\vec{\phi}}_{p\leftarrow q}.$$

These computations lead to two estimated protein interaction networks with expected probabilities of interactions taking values in $[0,1]$. We obtained binary protein interaction networks by thresholding these expected probabilities at ten different values. In terms of the two analyses described in Section 2.2, this amount to either (i)predicting physical interactions by thresholding the posterior expectations computed using blockmodel B and mixed membership map $\vec{\pi}s$, essentially a prediction task, or (ii) we de-noise the observed interactions Y using the blockmodel B and interaction-specific membership indicators Zs, essentially a de-noising task. We use the independent set of functional annotations from the gene ontology to decide which interactions are functionally meaningful; namely those between pairs of proteins that share at least one functional annotation (Myers et al., 2006). In this sense, between two models that suggest different sets of interactions as reliable, our evaluation assigns a higher score to the model that reveals *functionally relevant* interactions. Figure 13 shows the functional content of the original MIPS collection of physical interactions (point no.2), and of the collections of interactions computed using $(B,\Pi s)$, the light blue $(-\times)$ line, and using (B,Zs), the dark blue $(-+)$ line, thresholded at ten different levels—precision-recall curves. The posterior means of Πs provide a parsimonious representation for the MIPS collection, and lead to precise interaction estimates, in moderate amount ($-\times$ line). The posterior means of Zs provide a richer representation for the data, and describe most of the functional content of the MIPS collection with high precision ($-+$ line). Figure 13 also shows the functional content of the original MIPS collection (the yellow diamond). Most importantly, notice the estimated protein interaction

networks, that is, ex-es and crosses, corresponding to lower levels of recall feature a more precise functional content than the original. This means that the proposed latent block structure is helpful in summarizing the collection of interactions—by ranking them properly. On closer inspection, dense blocks of predicted interactions contain known functional predictions that were not in the MIPS collection, thus effectively improving the quality of the data that instantiate activity specific to few biological contexts, such as biopolymer catabolism and homeostasis. In conclusion, results suggest that MMB successfully reduces the dimensionality of the data, while revealing substantive information about the multiple functionality of proteins that can be used to inform subsequent analyses.

Table 3 provides more information about three instances of predicted interaction networks displayed in Figure 13; those corresponding the points annotated 1, 2, and 3. Specifically, the table shows a breakdown of the predicted (posterior) collections of interactions in each example network into the gene ontology categories. A count in the table corresponds to the fact that both proteins are annotated with the same GO functional category.[5]

In this application, the MMB learned information about (i) the mixed membership of objects to latent groups, and (ii) the connectivity patterns among latent groups. These estimates were useful in describing and summarizing the functional content of the MIPS collection of protein interactions. This suggests the use of MMB as a dimensionality reduction approach that may be useful for performing model-driven de-noising of new collections of interactions, such as those measured via high-throughput experiments.

5. Discussion

Modern probabilistic models for relational data analysis are rooted in the stochastic blockmodels for psychometric and sociological analysis, pioneered by Lorrain and White (1971) and by Holland and Leinhardt (1975). In statistics, this line of research has been extended in various contexts over the years (Fienberg et al., 1985; Wasserman and Pattison, 1996; Snijders, 2002; Hoff et al., 2002; Doreian et al., 2004). In machine learning, the related technique of Markov random networks (Frank and Strauss, 1986) have been used for link prediction (Taskar et al., 2003) and the traditional blockmodels have been extended to include nonparametric Bayesian priors (Kemp et al., 2004, 2006; Xu et al., 2006) and to integrate relations and text (McCallum et al., 2007).

There is a close relationship between the MMB and the latent space models (Hoff et al., 2002; Handcock et al., 2007). In the latent space models, the latent vectors are drawn from Gaussian distributions and the interaction data is drawn from a Gaussian with mean $\vec{\pi}_p{}' \mathbb{I} \vec{\pi}_q$. In the MMB, the marginal probability of an interaction takes a similar form, $\vec{\pi}_p{}' B \vec{\pi}_q$, where B is the matrix of probabilities of interactions for each pair of latent groups. Two major differences exist between these approaches. In MMB, the distribution over the latent vectors is a Dirichlet and the underlying data distribution is arbitrary—we have chosen Bernoulli. The posterior inference in latent space models (Hoff et al., 2002; Handcock et al., 2007) is carried out via MCMC sampling, while we have developed a scalable variational inference algorithm to analyze large network structures. (It would be interesting to develop a variational algorithm for the latent space models as well.) A number of well-designed numerical investigations and comparisons between variational EM and variants of MCMC have been performed in existing literature; for instance, see Buntine and Jakulin (2006),[6]

5. Note that, in GO, proteins are typically annotated to multiple functional categories.
6. See corresponding slides with additional results. (`http://www.hiit.fi/~buntine/dpca\_slides.pdf`)

#	GO Term	Description	Pred.	Tot.
1	GO:0043285	Biopolymer catabolism	561	17020
1	GO:0006366	Transcription from RNA polymerase II promoter	341	36046
1	GO:0006412	Protein biosynthesis	281	299925
1	GO:0006260	DNA replication	196	5253
1	GO:0006461	Protein complex assembly	191	11175
1	GO:0016568	Chromatin modification	172	15400
1	GO:0006473	Protein amino acid acetylation	91	666
1	GO:0006360	Transcription from RNA polymerase I promoter	78	378
1	GO:0042592	Homeostasis	78	5778
2	GO:0043285	Biopolymer catabolism	631	17020
2	GO:0006366	Transcription from RNA polymerase II promoter	414	36046
2	GO:0016568	Chromatin modification	229	15400
2	GO:0006260	DNA replication	226	5253
2	GO:0006412	Protein biosynthesis	225	299925
2	GO:0045045	Secretory pathway	151	18915
2	GO:0006793	Phosphorus metabolism	134	17391
2	GO:0048193	Golgi vesicle transport	128	9180
2	GO:0006352	Transcription initiation	121	1540
3	GO:0006412	Protein biosynthesis	277	299925
3	GO:0006461	Protein complex assembly	190	11175
3	GO:0009889	Regulation of biosynthesis	28	990
3	GO:0051246	Regulation of protein metabolism	28	903
3	GO:0007046	Ribosome biogenesis	10	21528
3	GO:0006512	Ubiquitin cycle	3	2211

Table 3: Breakdown of three example interaction networks into most represented gene ontology categories—see text for more details. The digit in the first column indicates the example network in Figure 13 that any given line refers to. The last two columns quote the number of predicted, and possible pairs for each GO term.

and Braun and McAuliffe (2007). We refer readers interested in the comparison between variational vs. MCMC to these resources.

The model decouples the observed connectivity patterns into two sources of variability, B, Πs, that are apparently in competition for explaining the data, possibly raising an identifiability issue. This is not the case, however, as the blockmodel B captures global/asymmetric relations, while the mixed membership vectors Πs capture local/symmetric relations. This difference practically eliminates the issue, unless there is no signal in the data to begin with.

A recurring question, which bears relevance to mixed membership models in general, is why we do not integrate out the single membership indicators—$(\vec{z}_{p\to q}, \vec{z}_{p\leftarrow q})$. While this may lead to computational efficiencies we would often lose interpretable quantities that are useful for making predictions, for de-noising new measurements, or for performing other tasks. In fact, the posterior distributions of such quantities typically carry substantive information about elements of the appli-

cation at hand. In the application to protein interaction networks of Section 4.3, for example, they encode the interaction-specific memberships of individual proteins to protein complexes.

In the relational setting, cross-validation is feasible if the blockmodel estimated on training data can be expected to hold on test data; for this to happen the network must be of reasonable size, so that we can expect members of each block to be in both training and test sets. In this setting, scheduling of variational updates is important; nested variational scheduling leads to efficient and parallelizable inference.

A limitation of our model can be best appreciated in a simulation setting. If we consider structural properties of the network MMB is capable of generating, we count a wide array of local and global connectivity patterns. But the model does not readily generate *hubs*, that is, nodes connected with a large number of directed or undirected connections, or networks with skewed degree distributions.

From a data analysis perspective, we speculate that the value of MMB in capturing substantive information about a problem will increase in semi-supervised setting—where, for example, information about the membership of genes to functional contexts is included in the form of prior distributions. In such a setting we may be interested in looking at the change between prior and posterior membership; a sharp change may signal biological phenomena worth investigating. We need not assume that the number of groups/blocks, K, is finite. It is possible, for example, to posit that the mixed-membership vectors are sampled form a stochastic process, in the nonparametric setting. To maintain mixed membership of nodes to groups/blocks in such setting, we need to sample them from a hierarchical Dirichlet process (Teh et al., 2006), rather than from a Dirichlet Process (Escobar and West, 1995).

MMB generalizes to two important cases. First, multiple data collections $Y_{1:M}$ on the same objects can be generated by the same latent vectors. This might be useful, for example, for simultaneously analyzing the relational measurements about esteem and disesteem, liking and disliking, positive influence and negative influence, praise and blame, for example, see Sampson (1968), or those about the collection of 17 relations measured by Bradley (1987). Second, in the MMB the data generating distribution is a Bernoulli, but B can be a matrix that parameterizes any kind of distribution. For example, technologies for measuring interactions between pairs of proteins such as mass spectrometry (Ho et al., 2002) and tandem affinity purification (Gavin et al., 2002) return a probabilistic assessment about the presence of interactions, thus setting the range of $Y(p,q)$ to $[0,1]$. This is not the case for the manually curated collection of interactions we analyze in Section 4.3.

6. Conclusions

In this paper we introduced mixed membership stochastic blockmodels, a novel class of latent variable models for relational data. These models provide exploratory tools for scientific analyses in applications where the observations can be represented as a collection of unipartite graphs. The nested variational inference algorithm is parallelizable and allows fast approximate inference on large graphs.

Acknowledgments

This work was partially supported by National Institutes of Health under Grant No. R01 AG023141-01, by the Office of Naval Research under Contracts N00014-02-1-0973 and 175-6343, by the National Science Foundation under Grants No. DMS-0240019, IIS-0218466, IIS-0745520and DBI-0546594, by the Pennsylvania Department of Health's Health Research Program under Grant No. 2001NF-Cancer Health Research Grant ME-01-739, and by the Department of Defense, all to Carnegie Mellon University. The authors would like to thank David Banks and Jim Berger at Duke University, Alan Karr at the National Institute of Statistical Sciences for insight and advice, and acknowledge generous support from the Statistical and Applied Mathematical Sciences Institute.

Appendix A. General Model Formulation

In general, mixed membership stochastic blockmodels can be specified in terms of assumptions at four levels: population, node, latent variable, and sampling scheme level.

A.1 Population Level

Assume that there are K classes or sub-populations in the population of interest. We denote by $f\ (\ Y(p,q) \mid B(g,h)\ )$ the probability distribution of the relation measured on the pair of nodes (p,q), where the *p-th* node is in the *h-th* sub-population, the *q-th* node is in the *h-th* sub-population, and $B(g,h)$ contains the relevant parameters. The indices i, j run in $1,\ldots,N$, and the indices g,h run in $1,\ldots,K$.

A.2 Node Level

The components of the membership vector $\vec{\pi}_p = [\vec{\pi}_p(1),\ldots,\vec{\pi}_p(k)]'$ encodes the mixed membership of the *n-th* node to the various sub-populations. The distribution of the observed response $Y(p,q)$ given the relevant, node-specific memberships, $(\vec{\pi}_p,\vec{\pi}_q)$, is then

$$Pr\ (\ Y(p,q) \mid \vec{\pi}_p,\vec{\pi}_q,B\ )\ \ =\ \ \sum_{g,h=1}^{K} \vec{\pi}_p(g)\ f(Y(p,q) \mid B(g,h))\ \vec{\pi}_q(h).$$

Conditional on the mixed memberships, the response edges y_{jnm} are independent of one another, both across distinct graphs and pairs of nodes.

A.3 Latent Variable Level

Assume that the mixed membership vectors $\vec{\pi}_{1:N}$ are realizations of a latent variable with distribution $D_{\vec{\alpha}}$, with parameter vector $\vec{\alpha}$. The probability of observing $Y(p,q)$, given the parameters, is then

$$Pr\ (\ Y(p,q) \mid \vec{\alpha},B\ )\ \ =\ \ \int Pr\ (\ Y(p,q) \mid \vec{\pi}_p,\vec{\pi}_q,B\ )\ D_{\vec{\alpha}}(d\vec{\pi}).$$

A.4 Sampling Scheme Level

Assume that the M independent replications of the relations measured on the population of nodes are independent of one another. The probability of observing the whole collection of graphs, $Y_{1:M}$, given the parameters, is then given by the following equation.

$$Pr\ (\ Y_{1:M} \mid \vec{\alpha},B\ ) = \prod_{m=1}^{M} \prod_{p,q=1}^{N} Pr\ (\ Y_m(p,q) \mid \vec{\alpha},B\ )\ .$$

Full model specifications immediately adapt to the different kinds of data, for example, multiple data types through the choice of f, or parametric or semi-parametric specifications of the prior on the number of clusters through the choice of a distribution for the πs, D_α.

Appendix B. Details of the Variational Approximation

Here we present more details about the derivation of the variational EM algorithm presented in Section 3. Furthermore, we address a setting where M replicates are available about the paired measurements, $G_{1:M} = (N, Y_{1:M})$, and relations $Y_m(p,q)$ take values into an arbitrary metric space according to $f\ (\ Y_m(p,q) \mid .. \ )$. An extension of the inference algorithm to address the case or multivariate relations, say J-dimensional, and multiple blockmodels $B_{1:J}$ each corresponding to a distinct relational response, can be derived with minor modifications of the derivations that follow.

B.1 Variational Expectation-Maximization

We begin by briefly summarizing the general strategy we intend to use. The approximate variant of EM we describe here is often referred to as *Variational EM* (Beal and Ghahramani, 2003). Recall that Y denotes the data. Rewrite $X = (\vec{\pi}_{1:N}, Z_{\rightarrow}, Z_{\leftarrow})$ for the latent variables, and $\Theta = (\vec{\alpha}, B)$ for the model's parameters. Briefly, it is possible to lower bound the likelihood, $p(Y|\Theta)$, making use of Jensen's inequality and of any distribution on the latent variables $q(X)$,

$$\begin{aligned} p(Y|\Theta) &= \log \int_{\mathcal{X}} p(Y,X|\Theta)\, dX \\ &= \log \int_{\mathcal{X}} q(X)\, \frac{p(Y,X|\Theta)}{q(X)}\, dX \qquad \text{(for any } q\text{)} \\ &\geq \int_{\mathcal{X}} q(X)\, \log \frac{p(Y,X|\Theta)}{q(X)}\, dX \qquad \text{(Jensen's)} \\ &= \mathbb{E}_q\, [\, \log p(Y,X|\Theta) - \log q(X)\,] \ =:\ \mathcal{L}(q,\Theta) \end{aligned}$$

In EM, the lower bound $\mathcal{L}(q,\Theta)$ is then iteratively maximized with respect to Θ, in the M step, and q in the E step (Dempster et al., 1977). In particular, at the *t-th* iteration of the E step we set

$$q^{(t)} = p(X|Y, \Theta^{(t-1)}), \tag{5}$$

that is, equal to the posterior distribution of the latent variables given the data and the estimates of the parameters at the previous iteration.

Unfortunately, we cannot compute the posterior in Equation 5 for the admixture of latent blocks model. Rather, we define a direct parametric approximation to it, $\tilde{q} = q_\Delta(X)$, which involves an extra set of *variational parameters*, Δ, and entails an approximate lower bound for the likelihood $\mathcal{L}_\Delta(q,\Theta)$. At the *t-th* iteration of the E step, we then minimize the Kullback-Leibler divergence between $q^{(t)}$ and $q^{(t)}_\Delta$, with respect to Δ, using the data.[7] The optimal parametric approximation is, in fact, a proper posterior as it depends on the data Y, although indirectly, $q^{(t)} \approx q^{(t)}_{\Delta^*(Y)}(X) = p(X|Y)$.

B.2 Lower Bound for the Likelihood

According to the mean-field theory (Jordan et al., 1999), one can approximate an intractable distribution such as the one defined by Equation (1) by a fully factored distribution $q(\vec{\pi}_{1:N}, Z^{\rightarrow}_{1:M}, Z^{\leftarrow}_{1:M})$

7. This is equivalent to maximizing the approximate lower bound for the likelihood, $\mathcal{L}_\Delta(q,\Theta)$, with respect to Δ.

defined as follows:

$$\begin{aligned} & q(\vec{\pi}_{1:N}, Z^{\rightarrow}_{1:M}, Z^{\leftarrow}_{1:M} | \vec{\gamma}_{1:N}, \Phi^{\rightarrow}_{1:M}, \Phi^{\leftarrow}_{1:M}) \\ = & \prod_p q_1(\vec{\pi}_p|\vec{\gamma}_p) \prod_m \prod_{p,q} \Big(q_2(\vec{z}^m_{p\rightarrow q}|\vec{\phi}^m_{p\rightarrow q}, 1)\; q_2(\vec{z}^m_{p\leftarrow q}|\vec{\phi}^m_{p\leftarrow q}, 1) \Big), \end{aligned}$$

where q_1 is a Dirichlet, q_2 is a multinomial, and $\Delta = (\vec{\gamma}_{1:N}, \Phi^{\rightarrow}_{1:M}, \Phi^{\leftarrow}_{1:M})$ represent the set of free *variational parameters* need to be estimated in the approximate distribution.

Minimizing the Kulback-Leibler divergence between this $q(\vec{\pi}_{1:N}, Z^{\rightarrow}_{1:M}, Z^{\leftarrow}_{1:M}|\Delta)$ and the original $p(\vec{\pi}_{1:N}, Z^{\rightarrow}_{1:M}, Z^{\leftarrow}_{1:M}$ defined by Equation (1) leads to the following approximate lower bound for the likelihood.

$$\begin{aligned} \mathcal{L}_\Delta(q,\Theta) &= \mathbb{E}_q\left[\log \prod_m \prod_{p,q} p_1(Y_m(p,q)|\vec{z}^m_{p\rightarrow q}, \vec{z}^m_{p\leftarrow q}, B)\right] \\ &+ \mathbb{E}_q\left[\log \prod_m \prod_{p,q} p_2(\vec{z}^m_{p\rightarrow q}|\vec{\pi}_p, 1)\right] + \mathbb{E}_q\left[\log \prod_m \prod_{p,q} p_2(\vec{z}^m_{p\leftarrow q}|\vec{\pi}_q, 1)\right] \\ &+ \mathbb{E}_q\left[\log \prod_p p_3(\vec{\pi}_p|\vec{\alpha})\right] - \mathbb{E}_q\left[\prod_p q_1(\vec{\pi}_p|\vec{\gamma}_p)\right] \\ &- \mathbb{E}_q\left[\log \prod_m \prod_{p,q} q_2(\vec{z}^m_{p\rightarrow q}|\vec{\phi}^m_{p\rightarrow q}, 1)\right] - \mathbb{E}_q\left[\log \prod_m \prod_{p,q} q_2(\vec{z}^m_{p\leftarrow q}|\vec{\phi}^m_{p\leftarrow q}, 1)\right]. \end{aligned}$$

Working on the single expectations leads to

$$\begin{aligned} \mathcal{L}_\Delta(q,\Theta) &= \sum_m \sum_{p,q} \sum_{g,h} \phi^m_{p\rightarrow q,g} \phi^m_{p\leftarrow q,h} \cdot f\left(Y_m(p,q), B(g,h)\right) \\ &+ \sum_m \sum_{p,q} \sum_g \phi^m_{p\rightarrow q,g}\left[\psi(\gamma_{p,g}) - \psi(\sum_g \gamma_{p,g})\right] \\ &+ \sum_m \sum_{p,q} \sum_h \phi^m_{p\leftarrow q,h}\left[\psi(\gamma_{p,h}) - \psi(\sum_h \gamma_{p,h})\right] \\ &+ \sum_p \log\Gamma(\sum_k \alpha_k) - \sum_{p,k} \log\Gamma(\alpha_k) + \sum_{p,k} (\alpha_k - 1)\left[\psi(\gamma_{p,k}) - \psi(\sum_k \gamma_{p,k})\right] \\ &- \sum_p \log\Gamma(\sum_k \gamma_{p,k}) + \sum_{p,k} \log\Gamma(\gamma_{p,k}) - \sum_{p,k} (\gamma_{p,k} - 1)\left[\psi(\gamma_{p,k}) - \psi(\sum_k \gamma_{p,k})\right] \\ &- \sum_m \sum_{p,q} \sum_g \phi^m_{p\rightarrow q,g} \log \phi^m_{p\rightarrow q,g} - \sum_m \sum_{p,q} \sum_h \phi^m_{p\leftarrow q,h} \log \phi^m_{p\leftarrow q,h} \end{aligned}$$

where

$$f\left(Y_m(p,q), B(g,h)\right) = Y_m(p,q) \log B(g,h) + \left(1 - Y_m(p,q)\right) \log\left(1 - B(g,h)\right);$$

m runs over $1,\ldots,M$; p,q run over $1,\ldots,N$; g,h,k run over $1,\ldots,K$; and $\psi(x)$ is the derivative of the log-gamma function, $\frac{d\log\Gamma(x)}{dx}$.

B.3 The Expected Value of the Log of a Dirichlet Random Vector

The computation of the lower bound for the likelihood requires us to evaluate $\mathbb{E}_q\left[\log \vec{\pi}_p\right]$ for $p = 1,\ldots,N$. Recall that the density of an exponential family distribution with natural parameter $\vec{\theta}$ can be written as

$$\begin{aligned} p(x|\alpha) &= h(x) \cdot c(\alpha) \cdot \exp\left\{\sum_k \theta_k(\alpha) \cdot t_k(x)\right\} \\ &= h(x) \cdot \exp\left\{\sum_k \theta_k(\alpha) \cdot t_k(x) - \log c(\alpha)\right\}. \end{aligned}$$

Omitting the node index p for convenience, we can rewrite the density of the Dirichlet distribution p_3 as an exponential family distribution,

$$p_3(\vec{\pi}|\vec{\alpha}) = \exp\left\{ \sum_k (\alpha_k - 1)\log(\pi_k) - \log\frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)} \right\},$$

with natural parameters $\theta_k(\vec{\alpha}) = (\alpha_k - 1)$ and natural sufficient statistics $t_k(\vec{\pi}) = \log(\pi_k)$. Let $c'(\vec{\theta}) = c(\alpha_1(\vec{\theta}), \ldots, \alpha_K(\vec{\theta}))$; using a well known property of the exponential family distributions (Schervish, 1995) we find that

$$\mathbb{E}_q\left[\log\pi_k\right] \;=\; \mathbb{E}_{\vec{\theta}}\left[\log t_k(x)\right] = \psi\left(\alpha_k\right) - \psi\left(\sum_k \alpha_k\right),$$

where $\psi(x)$ is the derivative of the log-gamma function, $\frac{d\log\Gamma(x)}{dx}$.

B.4 Variational E Step

The approximate lower bound for the likelihood $\mathcal{L}_\Delta(q,\Theta)$ can be maximized using exponential family arguments and coordinate ascent (Wainwright and Jordan, 2003).

Isolating terms containing $\phi^m_{p\to q,g}$ and $\phi^m_{p\leftarrow q,h}$ we obtain $\mathcal{L}_{\phi^m_{p\to q,g}}(q,\Theta)$ and $\mathcal{L}_{\phi^m_{p\to q,g}}(q,\Theta)$. The natural parameters $\vec{g}^m_{p\to q}$ and $\vec{g}^m_{p\leftarrow q}$ corresponding to the natural sufficient statistics $\log(\vec{z}^m_{p\to q})$ and $\log(\vec{z}^m_{p\leftarrow q})$ are functions of the other latent variables and the observations. We find that

$$\begin{aligned} g^m_{p\to q,g} &= \log\pi_{p,g} + \sum_h z^m_{p\leftarrow q,h}\cdot f\left(Y_m(p,q), B(g,h)\right), \\ g^m_{p\leftarrow q,h} &= \log\pi_{q,h} + \sum_g z^m_{p\to q,g}\cdot f\left(Y_m(p,q), B(g,h)\right), \end{aligned}$$

for all pairs of nodes (p,q) in the *m-th* network; where $g,h = 1,\ldots,K$, and

$$f\left(Y_m(p,q), B(g,h)\right) = Y_m(p,q)\log B(g,h) + \left(1 - Y_m(p,q)\right)\log\left(1 - B(g,h)\right).$$

This leads to the following updates for the variational parameters $(\vec{\phi}^m_{p\to q}, \vec{\phi}^m_{p\leftarrow q})$, for a pair of nodes (p,q) in the *m-th* network:

$$\begin{aligned} \hat{\phi}^m_{p\to q,g} &\propto e^{\,\mathbb{E}_q\left[g^m_{p\to q,g}\right]} \\ &= e^{\,\mathbb{E}_q\left[\log\pi_{p,g}\right]} \cdot e^{\,\sum_h \phi^m_{p\leftarrow q,h}\cdot\, \mathbb{E}_q\left[f\left(Y_m(p,q),B(g,h)\right)\right]} \\ &= e^{\,\mathbb{E}_q\left[\log\pi_{p,g}\right]} \cdot \prod_h \left(B(g,h)^{Y_m(p,q)} \cdot \left(1 - B(g,h)\right)^{1 - Y_m(p,q)} \right)^{\phi^m_{p\leftarrow q,h}}, \\ \hat{\phi}^m_{p\leftarrow q,h} &\propto e^{\,\mathbb{E}_q\left[g^m_{p\leftarrow q,h}\right]} \\ &= e^{\,\mathbb{E}_q\left[\log\pi_{q,h}\right]} \cdot e^{\,\sum_g \phi^m_{p\to q,g}\cdot\, \mathbb{E}_q\left[f\left(Y_m(p,q),B(g,h)\right)\right]} \\ &= e^{\,\mathbb{E}_q\left[\log\pi_{q,h}\right]} \cdot \prod_g \left(B(g,h)^{Y_m(p,q)} \cdot \left(1 - B(g,h)\right)^{1 - Y_m(p,q)} \right)^{\phi^m_{p\to q,g}}, \end{aligned}$$

for $g,h = 1,\ldots,K$. These estimates of the parameters underlying the distribution of the nodes' group indicators $\vec{\phi}^m_{p\to q}$ and $\vec{\phi}^m_{p\leftarrow q}$ need be normalized, to make sure $\sum_k \phi^m_{p\to q,k} = \sum_k \phi^m_{p\leftarrow q,k} = 1$.

Isolating terms containing $\gamma_{p,k}$ we obtain $\mathcal{L}_{\gamma_{p,k}}(q,\Theta)$. Setting $\frac{\partial \mathcal{L}_{\gamma_{p,k}}}{\partial \gamma_{p,k}}$ equal to zero and solving for $\gamma_{p,k}$ yields:

$$\hat{\gamma}_{p,k} = \alpha_k + \sum_m \sum_q \phi^m_{p\to q,k} + \sum_m \sum_q \phi^m_{p\leftarrow q,k},$$

for all nodes $p \in \mathcal{P}$ and $k = 1,\ldots,K$.

The *t-th* iteration of the variational E step is carried out for fixed values of $\Theta^{(t-1)} = (\vec{\alpha}^{(t-1)}, B^{(t-1)})$, and finds the optimal approximate lower bound for the likelihood $\mathcal{L}_{\Delta^*}(q,\Theta^{(t-1)})$.

B.5 Variational M Step

The optimal lower bound $\mathcal{L}_{\Delta^*}(q^{(t-1)},\Theta)$ provides a tractable surrogate for the likelihood at the *t-th* iteration of the variational M step. We derive empirical Bayes estimates for the hyper-parameters Θ that are based upon it.[8] That is, we maximize $\mathcal{L}_{\Delta^*}(q^{(t-1)},\Theta)$ with respect to Θ, given expected sufficient statistics computed using $\mathcal{L}_{\Delta^*}(q^{(t-1)},\Theta^{(t-1)})$.

Isolating terms containing $\vec{\alpha}$ we obtain $\mathcal{L}_{\vec{\alpha}}(q,\Theta)$. Unfortunately, a closed form solution for the approximate maximum likelihood estimate of $\vec{\alpha}$ does not exist (Blei et al., 2003). We can produce a Newton-Raphson method that is linear in time, where the gradient and Hessian for the bound $\mathcal{L}_{\vec{\alpha}}$ are

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\vec{\alpha}}}{\partial \alpha_k} &= N\left(\psi\left(\sum_k \alpha_k\right) - \psi(\alpha_k)\right) + \sum_p \left(\psi(\gamma_{p,k}) - \psi\left(\sum_k \gamma_{p,k}\right)\right), \\
\frac{\partial \mathcal{L}_{\vec{\alpha}}}{\partial \alpha_{k_1}\alpha_{k_2}} &= N\left(\mathbb{I}_{(k_1=k_2)}\cdot\psi'(\alpha_{k_1}) - \psi'\left(\sum_k \alpha_k\right)\right).
\end{aligned}$$

Isolating terms containing B we obtain $\mathcal{L}_B$, whose approximate maximum is

$$\hat{B}(g,h) = \frac{1}{M}\sum_m \left(\frac{\sum_{p,q} Y_m(p,q)\cdot \phi^m_{p\to qg}\phi^m_{p\leftarrow qh}}{(1-\rho)\cdot\sum_{p,q}\phi^m_{p\to qg}\phi^m_{p\leftarrow qh}}\right),$$

for every index pair $(g,h) \in [1,K]\times[1,K]$.

In Section 2.1 we introduced an extra parameter, ρ, to control the relative importance of presence and absence of interactions in likelihood, that is, the score that informs inference and estimation. Isolating terms containing ρ we obtain $\mathcal{L}_\rho$. We may then estimate the sparsity parameter ρ by

$$\hat{\rho} = \frac{1}{M}\sum_m \left(\frac{\sum_{p,q}\left(1 - Y_m(p,q)\right)\cdot\left(\sum_{g,h}\phi^m_{p\to qg}\phi^m_{p\leftarrow qh}\right)}{\sum_{p,q}\sum_{g,h}\phi^m_{p\to qg}\phi^m_{p\leftarrow qh}}\right).$$

Alternatively, we can fix ρ prior to the analysis; the density of the interaction matrix is estimated with $\hat{d} = \sum_{m,p,q} Y_m(p,q)/(N^2M)$, and the sparsity parameter is set to $\tilde{\rho} = (1-\hat{d})$. This latter estimator attributes all the information in the non-interactions to the point mass, that is, to latent sources other than the block model B or the mixed membership vectors $\vec{\pi}_{1:N}$. It does, however, provide a quick recipe to reduce the computational burden during exploratory analyses.[9]

8. We could term these estimates *pseudo* empirical Bayes estimates, since they maximize an approximate lower bound for the likelihood, $\mathcal{L}_{\Delta^*}$.

9. Note that $\tilde{\rho} = \hat{\rho}$ in the case of single membership. In fact, that implies $\phi^m_{p\to qg} = \phi^m_{p\leftarrow qh} = 1$ for some (g,h) pair, for any (p,q) pair.

References

E. M. Airoldi. Getting started in probabilistic graphical models. *PLoS Computational Biology*, 3 (12):e252, 2007.

E. M. Airoldi, D. M. Blei, E. P. Xing, and S. E. Fienberg. A latent mixed-membership model for relational data. In *ACM SIGKDD Workshop on Link Discovery: Issues, Approaches and Applications*, 2005.

E. M. Airoldi, S. E. Fienberg, and E. P. Xing. Mixed membership analysis of expression studies—attribute data. Manuscript, 2007. URL `http://arxiv.org/abs/0711.2520/`.

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland, 4th edition, 2002.

M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubinand, and G. Sherlock. Gene ontology: Tool for the unification of biology. The gene ontology consortium. *Nature Genetics*, 25(1):25–29, 2000.

M. J. Beal and Z. Ghahramani. The variational Bayesian EM algorithm for incomplete data: With application to scoring graphical model structures. In J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, editors, *Bayesian Statistics*, volume 7, pages 453–464. Oxford University Press, 2003.

L. Berkman, B. H. Singer, and K. Manton. Black/white differences in health status and mortality among the elderly. *Demography*, 26(4):661–678, 1989.

C. Bishop, D. Spiegelhalter, and J. Winn. VIBES: A variational inference engine for Bayesian networks. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 777–784. MIT Press, Cambridge, MA, 2003.

D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

R. T. Bradley. *Charisma and Social Structure*. Paragon House, 1987.

M. Braun and J. McAuliffe. Variational inference for large-scale models of discrete choice. Manuscript, 2007. URL `http://arxiv.org/abs/0712.2526/`.

R. L. Breiger, S. A. Boorman, and P. Arabie. An algorithm for clustering relational data with applications to social network analysis and comparison to multidimensional scaling. *Journal of Mathematical Psychology*, 12:328–383, 1975.

W. L. Buntine and A. Jakulin. Discrete components analysis. In C. Saunders, M. Grobelnik, S. Gunn, and J. Shawe-Taylor, editors, *Subspace, Latent Structure and Feature Selection Techniques*. Springer-Verlag, 2006. URL `http://arxiv.org/abs/math.ST/0604410/`.

G. B. Davis and K. M. Carley. Clearing the FOG: Fuzzy, overlapping groups for social networks. Manuscript, 2006.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

P. Doreian, V. Batagelj, and A. Ferligoj. *Generalized Blockmodeling*. Cambridge University Press, 2004.

P. Doreian, V. Batagelj, and A. Ferligoj. Discussion of "Model-based clustering for social networks". *Journal of the Royal Statistical Society, Series A*, 170, 2007.

E. A. Erosheva. *Grade of Membership and Latent Structure Models with Application to Disability Survey Data*. PhD thesis, Carnegie Mellon University, Department of Statistics, 2002.

E. A. Erosheva and S. E. Fienberg. Bayesian mixed membership models for soft clustering and classification. In C. Weihs and W. Gaul, editors, *Classification—The Ubiquitous Challenge*, pages 11–26. Springer-Verlag, 2005.

M. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90:577–588, 1995.

S. E. Fienberg, M. M. Meyer, and S. Wasserman. Statistical analysis of multiple sociometric relations. *Journal of the American Statistical Association*, 80:51–67, 1985.

O. Frank and D. Strauss. Markov graphs. *Journal of the American Statistical Association*, 81: 832–842, 1986.

A. C. Gavin, M. Bosche, R. Krause, P. Grandi, M. Marzioch, A. Bauer, J. Schultz, and et. al. Functional organization of the yeast proteome by systematic analysis of protein complexes. *Nature*, 415:141–147, 2002.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, 2004.

M. S. Handcock, A. E. Raftery, and J. M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society, Series A*, 170:1–22, 2007.

K. M. Harris, F. Florey, J. Tabor, P. S. Bearman, J. Jones, and R. J. Udry. The national longitudinal study of adolescent health: research design. Technical report, Caorlina Population Center, University of North Carolina, Chapel Hill, 2003.

Y. Ho, A. Gruhler, A. Heilbut, G. D. Bader, L. Moore, S. L. Adams, A. Millar, P. Taylor, K. Bennett, and K. Boutilier et. al. Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, 415:180–183, 2002.

P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098, 2002.

P. W. Holland and S. Leinhardt. Local structure in social networks. In D. Heise, editor, *Sociological Methodology*, pages 1–45. Jossey-Bass, 1975.

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

C. Joutard, E. M. Airoldi, S. E. Fienberg, and T. M. Love. Discovery of latent patterns with hierarchical bayesian mixed-membership models and the issue of model choice. In *Data Mining Patterns, New Methods and Applications*, 2007. Forthcoming.

C. Kemp, T. L. Griffiths, and J. B. Tenenbaum. Discovering latent classes in relational data. Technical Report AI Memo 2004-019, MIT, 2004.

C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, T. Punna, J. M. Peregrin-Alvarez, M. Shales, X. Zhang, M. Davey, M. D. Robinson, A. Paccanaro, J. E. Bray, A. Sheung, B. Beattie, D. P. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M. M. Canete, J. Vlasblom, S. Wu, C. Orsi, S. R. Collins, S. Chandran, R. Haw, J. J. Rilstone, K. Gandi, N. J. Thompson, G. Musso, P. St Onge, S. Ghanny, M. H. Y. Lam, G. Butland, A. M. Altaf-Ul, S. Kanaya, A. Shilatifard, E. O'Shea, J. S. Weissman, C. J. Ingles, T. R. Hughes, J. Parkinson, M. Gerstein, S. J. Wodak, A. Emili, and J. F. Greenblatt. Global landscape of protein complexes in the yeast Saccharomyces Cerevisiae. *Nature*, 440 (7084):637–643, 2006.

F.-F. Li and P. Perona. A Bayesian hierarchical model for learning natural scene categories. *IEEE Computer Vision and Pattern Recognition*, 2005.

F. Lorrain and H. C. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1:49–80, 1971.

A. McCallum, X. Wang, and N. Mohanty. Joint group and topic discovery from relations and text. In *Statistical Network Analysis: Models, Issues and New Directions*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

H. W. Mewes, C. Amid, R. Arnold, D. Frishman, U. Guldener, and et. al. Mips: analysis and annotation of proteins from whole genomes. *Nucleic Acids Research*, 32:D41–44, 2004.

T. Minka. Estimating a Dirichlet distribution. Manuscript, 2003.

T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Uncertainty in Artificial Intelligence*, 2002.

C. L. Myers, D. A. Barret, M. A. Hibbs, C. Huttenhower, and O. G. Troyanskaya. Finding function: An evaluation framework for functional genomics. *BMC Genomics*, 7(187), 2006.

J. K. Pritchard, M. Stephens, N. A. Rosenberg, and P. Donnelly. Association mapping in structured populations. *American Journal of Human Genetics*, 67:170–181, 2000.

F. S. Sampson. *A Novitiate in a Period of Change: An Experimental and Case Study of Social Relationships*. PhD thesis, Cornell University, 1968.

Mark J. Schervish. *Theory of Statistics*. Springer, 1995.

T. A. B. Snijders. Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure*, 2002.

T. A. B. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100, 1997.

B. Taskar, M. F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Neural Information Processing Systems 15*, 2003.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Y. W. Teh, D. Newman, and M. Welling. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 19, 2007.

R. J. Udry. The national longitudinal study of adolescent health: (add health) waves i and ii, 1994–1996; wave iii 2001–2002. Technical report, Caorlina Population Center, University of North Carolina, Chapel Hill, 2003.

C. T. Volinsky and A. E. Raftery. Bayesian information criterion for censored survival models. *Biometrics*, 56:256–262, 2000.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families and variational inference. Technical Report 649, Department of Statistics, University of California, Berkeley, 2003.

Y. J. Wang and G. Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82:8–19, 1987.

S. Wasserman and P. Pattison. Logit models and logistic regression for social networks: I. an introduction to markov graphs and p^*. *Psychometrika*, 61:401–425, 1996.

E. P. Xing, M. I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Uncertainty in Artificial Intelligence*, volume 19, 2003.

Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. Infinite hidden relational models. In *Uncertainty in Artificial Intelligence*, 2006.

Journal of Machine Learning Research 9 (2008) 2015-2033 Submitted 1/08; Revised 5/08; Published 9/08

Consistency of Random Forests and Other Averaging Classifiers

Gérard Biau GERARD.BIAU@UPMC.FR
LSTA & LPMA
Université Pierre et Marie Curie – Paris VI
Boîte 158, 175 rue du Chevaleret
75013 Paris, France

Luc Devroye LUC@CS.MCGILL.CA
School of Computer Science
McGill University
Montreal, Canada H3A 2K6

Gábor Lugosi LUGOSI@UPF.ES
ICREA and Department of Economics
Pompeu Fabra University
Ramon Trias Fargas 25-27
08005 Barcelona, Spain

Editor: Peter Bartlett

Abstract

In the last years of his life, Leo Breiman promoted random forests for use in classification. He suggested using averaging as a means of obtaining good discrimination rules. The base classifiers used for averaging are simple and randomized, often based on random samples from the data. He left a few questions unanswered regarding the consistency of such rules. In this paper, we give a number of theorems that establish the universal consistency of averaging rules. We also show that some popular classifiers, including one suggested by Breiman, are not universally consistent.

Keywords: random forests, classification trees, consistency, bagging

This paper is dedicated to the memory of Leo Breiman.

1. Introduction

Ensemble methods, popular in machine learning, are learning algorithms that construct a set of many individual classifiers (called base learners) and combine them to classify new data points by taking a weighted or unweighted vote of their predictions. It is now well-known that ensembles are often much more accurate than the individual classifiers that make them up. The success of ensemble algorithms on many benchmark data sets has raised considerable interest in understanding why such methods succeed and identifying circumstances in which they can be expected to produce good results. These methods differ in the way the base learner is fit and combined. For example, bagging (Breiman, 1996) proceeds by generating bootstrap samples from the original data set, constructing a classifier from each bootstrap sample, and voting to combine. In boosting (Freund and Schapire, 1996) and arcing algorithms (Breiman, 1998) the successive classifiers are constructed by giving increased weight to those points that have been frequently misclassified, and the classifiers are combined using weighted voting. On the other hand, random split selection (Dietterich, 2000)

grows trees on the original data set. For a fixed number S, at each node, S best splits (in terms of minimizing deviance) are found and the actual split is randomly and uniformly selected from them. For a comprehensive review of ensemble methods, we refer the reader to Dietterich (2000a) and the references therein.

Breiman (2001) provides a general framework for tree ensembles called "random forests". Each tree depends on the values of a random vector sampled independently and with the same distribution for all trees. Thus, a random forest is a classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. Algorithms for inducing a random forest were first developed by Breiman and Cutler, and "Random Forests" is their trademark. The web page

http://www.stat.berkeley.edu/users/breiman/RandomForests

provides a collection of downloadable technical reports, and gives an overview of random forests as well as comments on the features of the method.

Random forests have been shown to give excellent performance on a number of practical problems. They work fast, generally exhibit a substantial performance improvement over single tree classifiers such as CART, and yield generalization error rates that compare favorably to the best statistical and machine learning methods. In fact, random forests are among the most accurate general-purpose classifiers available (see, for example, Breiman, 2001).

Different random forests differ in how randomness is introduced in the tree building process, ranging from extreme random splitting strategies (Breiman, 2000; Cutler and Zhao, 2001) to more involved data-dependent strategies (Amit and Geman, 1997; Breiman, 2001; Dietterich, 2000). As a matter of fact, the statistical mechanism of random forests is not yet fully understood and is still under active investigation. Unlike single trees, where consistency is proved letting the number of observations in each terminal node become large (Devroye, Györfi, and Lugosi, 1996, Chapter 20), random forests are generally built to have a small number of cases in each terminal node. Although the mechanism of random forest algorithms appears simple, it is difficult to analyze and remains largely unknown. Some attempts to investigate the driving force behind consistency of random forests are by Breiman (2000, 2004) and Lin and Jeon (2006), who establish a connection between random forests and adaptive nearest neighbor methods. Meinshausen (2006) proved consistency of certain random forests in the context of so-called quantile regression.

In this paper we offer consistency theorems for various versions of random forests and other randomized ensemble classifiers. In Section 2 we introduce a general framework for studying classifiers based on averaging randomized base classifiers. We prove a simple but useful proposition showing that averaged classifiers are consistent whenever the base classifiers are.

In Section 3 we prove consistency of two simple random forest classifiers, the *purely random forest* (suggested by Breiman as a starting point for study) and the *scale-invariant random forest* classifiers.

In Section 4 it is shown that averaging may convert inconsistent rules into consistent ones.

In Section 5 we briefly investigate consistency of bagging rules. We show that, in general, bagging preserves consistency of the base rule and it may even create consistent rules from inconsistent ones. In particular, we show that if the bootstrap samples are sufficiently small, the bagged version of the 1-nearest neighbor classifier is consistent.

Finally, in Section 6 we consider random forest classifiers based on randomized, greedily grown tree classifiers. We argue that some greedy random forest classifiers, including Breiman's random forest classifier, are inconsistent and suggest a consistent greedy random forest classifier.

2. Voting and Averaged Classifiers

Let $(X,Y),(X_1,Y_1),\ldots,(X_n,Y_n)$ be i.i.d. pairs of random variables such that X (the so-called *feature vector*) takes its values in $\mathbb{R}^d$ while Y (the *label*) is a binary $\{0,1\}$-valued random variable. The joint distribution of (X,Y) is determined by the marginal distribution μ of X (i.e., $\mathbb{P}\{X \in A\} = \mu(A)$ for all Borel sets $A \subset \mathbb{R}^d$) and the *a posteriori* probability $\eta : \mathbb{R}^d \to [0,1]$ defined by

$$\eta(x) = \mathbb{P}\{Y = 1 | X = x\} \ .$$

The collection $(X_1,Y_1),\ldots,(X_n,Y_n)$ is called the *training data*, and is denoted by D_n. A classifier g_n is a binary-valued function of X and D_n whose probability of error is defined by

$$L(g_n) = \mathbb{P}_{(X,Y)}\{g_n(X,D_n) \neq Y\}$$

where $\mathbb{P}_{(X,Y)}$ denotes probability with respect to the pair (X,Y) (i.e., conditional probability, given D_n). For brevity, we write $g_n(X) = g_n(X,D_n)$. It is well-known (see, for example, Devroye, Györfi, and Lugosi, 1996) that the classifier that minimizes the probability of error, the so-called *Bayes classifier* is $g^*(x) = \mathbb{1}_{\{\eta(x) \geq 1/2\}}$. The risk of g^* is called the Bayes risk: $L^* = L(g^*)$.

A sequence $\{g_n\}$ of classifiers is *consistent* for a certain distribution of (X,Y) if $L(g_n) \to L^*$ in probability.

In this paper we investigate classifiers that calculate their decisions by taking a majority vote over *randomized classifiers*. A randomized classifier may use a random variable Z to calculate its decision. More precisely, let $\mathcal{Z}$ be some measurable space and let Z take its values in $\mathcal{Z}$. A randomized classifier is an arbitrary function of the form $g_n(X,Z,D_n)$, which we abbreviate by $g_n(X,Z)$. The probability of error of g_n becomes

$$L(g_n) = \mathbb{P}_{(X,Y),Z}\{g_n(X,Z,D_n) \neq Y\} = \mathbb{P}\{g_n(X,Z,D_n) \neq Y | D_n\} \ .$$

The definition of consistency remains the same by augmenting the probability space appropriately to include the randomization.

Given any randomized classifier, one may calculate the classifier for various draws of the randomizing variable Z. It is then a natural idea to define an averaged classifier by taking a majority vote among the obtained random classifiers. Assume that $Z_1,\ldots,Z_m$ are identically distributed draws of the randomizing variable, having the same distribution as Z. Throughout the paper, we assume that $Z_1,\ldots,Z_m$ are independent, conditionally on X, Y, and D_n. Letting $Z^m = (Z_1,\ldots,Z_m)$, one may define the corresponding *voting classifier* by

$$g_n^{(m)}(x,Z^m,D_n) = \begin{cases} 1 & \text{if } \frac{1}{m}\sum_{j=1}^m g_n(x,Z_j,D_n) \geq \frac{1}{2} \ , \\ 0 & \text{otherwise.} \end{cases}$$

By the strong law of large numbers, for any fixed x and D_n for which $\mathbb{P}_Z\{g_n(x,Z,D_n) = 1\} \neq 1/2$, we have almost surely $\lim_{m\to\infty} g_n^{(m)}(x,Z^m,D_n) = \bar{g}_n(x,D_n)$, where $\bar{g}_n(x,D_n) = \bar{g}_n(x) = \mathbb{1}_{\{\mathbb{E}_Z g_n(x,Z) \geq 1/2\}}$

is a (non-randomized) classifier that we call the *averaged classifier.* (Here $\mathbb{P}_Z$ and $\mathbb{E}_Z$ denote probability and expectation with respect to the randomizing variable Z, that is, conditionally on X, Y, and D_n.)

$\bar{g}_n$ may be interpreted as an idealized version of the classifier $g_n^{(m)}$ that draws many independent copies of the randomizing variable Z and takes a majority vote over the resulting classifiers.

Our first result states that consistency of a randomized classifier is preserved by averaging.

Proposition 1 *Assume that the sequence $\{g_n\}$ of randomized classifiers is consistent for a certain distribution of (X,Y). Then the voting classifier $g_n^{(m)}$ (for any value of m) and the averaged classifier $\bar{g}_n$ are also consistent.*

Proof Consistency of $\{g_n\}$ is equivalent to saying that $\mathbb{E}L(g_n) = \mathbb{P}\{g_n(X,Z) \neq Y\} \to L^*$. In fact, since $\mathbb{P}\{g_n(X,Z) \neq Y | X = x\} \geq \mathbb{P}\{g^*(X) \neq Y | X = x\}$ for all $x \in \mathbb{R}^d$, consistency of $\{g_n\}$ means that for μ-almost all x,

$$\mathbb{P}\{g_n(X,Z) \neq Y | X = x\} \to \mathbb{P}\{g^*(X) \neq Y | X = x\} = \min(\eta(x), 1-\eta(x)) \; .$$

Without loss of generality, assume that $\eta(x) > 1/2$. (In the case of $\eta(x) = 1/2$ any classifier has a conditional probability of error $1/2$ and there is nothing to prove.) Then $\mathbb{P}\{g_n(X,Z) \neq Y | X = x\} = (2\eta(x)-1)\mathbb{P}\{g_n(x,Z) = 0\} + 1 - \eta(x)$, and by consistency we have $\mathbb{P}\{g_n(x,Z) = 0\} \to 0$.

To prove consistency of the voting classifier $g_n^{(m)}$, it suffices to show that $\mathbb{P}\{g_n^{(m)}(x,Z^m) = 0\} \to 0$ for μ-almost all x for which $\eta(x) > 1/2$. However,

$$\begin{aligned}
\mathbb{P}\{g_n^{(m)}(x,Z^m) = 0\} &= \mathbb{P}\left\{ (1/m) \sum_{j=1}^m \mathbb{1}_{\{g_n(x,Z_j)=0\}} > 1/2 \right\} \\
&\leq 2\mathbb{E}\left[(1/m) \sum_{j=1}^m \mathbb{1}_{\{g_n(x,Z_j)=0\}} \right] \\
&\qquad \text{(by Markov's inequality)} \\
&= 2\mathbb{P}\{g_n(x,Z) = 0\} \to 0 \; .
\end{aligned}$$

Consistency of the averaged classifier is proved by a similar argument.

□

3. Random Forests

Random forests, introduced by Breiman, are averaged classifiers in the sense defined in Section 2.

Formally, a random forest with m trees is a classifier consisting of a collection of randomized base tree classifiers $g_n(x,Z_1), \ldots, g_n(x,Z_m)$ where $Z_1, \ldots, Z_m$ are identically distributed random vectors, independent conditionally on X, Y, and D_n.

The randomizing variable is typically used to determine how the successive cuts are performed when building the tree such as selection of the node and the coordinate to split, as well as the position of the split. The random forest classifier takes a majority vote among the random tree classifiers. If m is large, the random forest classifier is well approximated by the averaged classifier

$\overline{g}_n(x) = \mathbb{1}_{\{\mathbb{E}_Z g_n(x,Z) \geq 1/2\}}$. For brevity, we state most results of this paper for the averaged classifier only, though by Proposition 1 various results remain true for the voting classifier $g_n^{(m)}$ as well.

In this section we analyze a simple random forest already considered by Breiman (2000), which we call the *purely random forest*.

The random tree classifier $g_n(x,Z)$ is constructed as follows. Assume, for simplicity, that μ is supported on $[0,1]^d$. All nodes of the tree are associated with rectangular cells such that at each step of the construction of the tree, the collection of cells associated with the leaves of the tree (i.e., external nodes) forms a partition of $[0,1]^d$. The root of the random tree is $[0,1]^d$ itself. At each step of the construction of the tree, a leaf is chosen uniformly at random. The split variable J is then selected uniformly at random from the d candidates $x^{(1)},\ldots,x^{(d)}$. Finally, the selected cell is split along the randomly chosen variable at a random location, chosen according to a uniform random variable on the length of the chosen side of the selected cell. The procedure is repeated k times where $k \geq 1$ is a deterministic parameter, fixed beforehand by the user, and possibly depending on n.

The randomized classifier $g_n(x,Z)$ takes a majority vote among all Y_i for which the corresponding feature vector X_i falls in the same cell of the random partition as x. (For concreteness, break ties in favor of the label 1.)

The purely random forest classifier is a radically simplified version of random forest classifiers used in practice. The main simplification lies in the fact that recursive cell splits do not depend on the labels $Y_1,\ldots,Y_n$. The next theorem mainly serves as an illustration of how the consistency problem of random forest classifiers may be attacked. More involved versions of random forest classifiers are discussed in subsequent sections.

Theorem 2 *Assume that the distribution of X is supported on $[0,1]^d$. Then the purely random forest classifier $\overline{g}_n$ is consistent whenever $k \to \infty$ and $k/n \to 0$ as $k \to \infty$.*

Proof By Proposition 1 it suffices to prove consistency of the randomized base tree classifier g_n. To this end, we recall a general consistency theorem for partitioning classifiers proved in (Devroye, Györfi, and Lugosi, 1996, Theorem 6.1). According to this theorem, g_n is consistent if both $\mathrm{diam}(A_n(X,Z)) \to 0$ in probability and $N_n(X,Z) \to \infty$ in probability, where $A_n(x,Z)$ is the rectangular cell of the random partition containing x and

$$N_n(x,Z) = \sum_{i=1}^{n} \mathbb{1}_{\{X_i \in A_n(x,Z)\}}$$

is the number of data points falling in the same cell as x.

First we show that $N_n(X,Z) \to \infty$ in probability. Consider the random tree partition defined by Z. Observe that the partition has $k+1$ rectangular cells, say $A_1,\ldots,A_{k+1}$. Let $N_1,\ldots,N_{k+1}$ denote the number of points of $X,X_1,\ldots,X_n$ falling in these $k+1$ cells. Let $S = \{X,X_1,\ldots,X_n\}$ denote the set of positions of these $n+1$ points. Since these points are independent and identically distributed, fixing the *set* S (but not the order of the points) and Z, the conditional probability that X falls in the i-th cell equals $N_i/(n+1)$. Thus, for every fixed $t > 0$,

$$\begin{aligned}
\mathbb{P}\{N_n(X,Z) < t\} &= \mathbb{E}\left[\mathbb{P}\{N_n(X,Z) < t | S, Z\}\right] \\
&= \mathbb{E}\left[\sum_{i:N_i<t} \frac{N_i}{n+1}\right] \leq (t-1)\frac{k+1}{n+1}
\end{aligned}$$

which converges to zero by our assumption on k.

It remains to show that $\mathrm{diam}(A_n(X,Z)) \to 0$ in probability. To this aim, let $V_n = V_n(x,Z)$ be the size of the first dimension of the rectangle containing x. Let $T_n = T_n(x,Z)$ be the number of times that the box containing x is split when we construct the random tree partition.

Let K_n be binomial $(T_n, 1/d)$, representing the number of times the box containing x is split along the first coordinate.

Clearly, it suffices to show that $V_n(x,Z) \to 0$ in probability for μ-almost all x, so it is enough to show that for all x, $\mathbb{E}[V_n(x,Z)] \to 0$. Observe that if $U_1, U_2, \ldots$ are independent uniform $[0,1]$, then

$$\begin{aligned}
\mathbb{E}[V_n(x,Z)] &\leq \mathbb{E}\left[\mathbb{E}\left[\prod_{i=1}^{K_n} \max(U_i, 1-U_i) \,\middle|\, K_n\right]\right] \\
&= \mathbb{E}\left[\mathbb{E}[\max(U_1, 1-U_1)]^{K_n}\right] \\
&= \mathbb{E}\left[(3/4)^{K_n}\right] \\
&= \mathbb{E}\left[\left(1 - \frac{1}{d} + \frac{3}{4d}\right)^{T_n}\right] \\
&= \mathbb{E}\left[\left(1 - \frac{1}{4d}\right)^{T_n}\right].
\end{aligned}$$

Thus, it suffices to show that $T_n \to \infty$ in probability. To this end, note that the partition tree is statistically related to a random binary search tree with $k+1$ external nodes (and thus k internal nodes). Such a tree is obtained as follows. Initially, the root is the sole external node, and there are no internal nodes. Select an external node uniformly at random, make it an internal node and give it two children, both external. Repeat until we have precisely k internal nodes and $k+1$ external nodes. The resulting tree is the random binary search tree on k internal nodes (see Devroye 1988 and Mahmoud 1992 for more equivalent constructions of random binary search trees). It is known that all levels up to $\ell = \lfloor 0.37 \log k \rfloor$ are full with probability tending to one as $k \to \infty$ (Devroye, 1986). The last full level F_n is called the fill-up level. Clearly, the partition tree has this property. Therefore, we know that all final cells have been cut at least ℓ times and therefore $T_n \geq \ell$ with probability converging to 1. This concludes the proof of Theorem 3.1. □

Remark 3 *We observe that the largest first dimension among external nodes does not tend to zero in probability except for $d = 1$. For $d \geq 2$, it tends to a limit random variable that is not atomic at zero (this can be shown using the theory of branching processes). Thus the proof above could not have used the uniform smallness of all cells. Despite the fact that the random partition contains some cells of huge diameter of non-shrinking size, the rule based on it is consistent.*

Next we consider a scale-invariant version of the purely random forest classifier. In this variant the root cell is the entire feature space and the random tree is grown up to k cuts. The leaf cell to cut and the direction J in which the cell is cut are chosen uniformly at random, exactly as in the purely random forest classifier. The only difference is that the position of the cut is now chosen in a data-based manner: if the cell to be cut contains N of the data points $X, X_1, \ldots, X_n$, then a random index I is chosen uniformly from the set $\{0, 1, \ldots, N\}$ and the cell is cut so that, when ordered by their J-th components, the points with the I smallest values fall in one of the subcells and the rest in

the other. To avoid ties, we assume that the distribution of X has non-atomic marginals. In this case the random tree is well-defined with probability one. Just like before, the associated classifier takes a majority vote over the labels of the data points falling in the same cell as X. The *scale-invariant random forest* classifier is defined as the corresponding averaged classifier.

Theorem 4 *Assume that the distribution of X has non-atomic marginals in $\mathbb{R}^d$. Then the scale-invariant random forest classifier $\bar{g}_n$ is consistent whenever $k \to \infty$ and $k/n \to 0$ as $k \to \infty$.*

Proof Once again, we may use Proposition 1 and (Devroye, Györfi, and Lugosi, 1996, Theorem 6.1) to prove consistency of the randomized base tree classifier g_n. The proof of the fact that $N_n(X,Z) \to \infty$ in probability is the same as in Theorem 2.

To show that $\mathrm{diam}(A_n(X,Z)) \to 0$ in probability, we begin by noting that, just as in the case of the purely random forest classifier, the partition tree is equivalent to a binary search tree, and therefore with probability converging to one, all final cells have been cut at least $\ell = \lfloor 0.37 \log k \rfloor$ times.

Since the classification rule is scale-invariant, we may assume, without loss of generality, that the distribution of X is concentrated on the unit cube $[0,1]^d$.

Let n_i denote the cardinality of the i-th cell in the partition, $1 \le i \le k+1$, where the cardinality of a cell C is $|C \cap \{X, X_1, \ldots, X_n\}|$. Thus, $\sum_{i=1}^{k+1} n_i = n+1$. Let V_i be the first dimension of the i-th cell. Let $V(X)$ be the first dimension of the cell that contains X. Clearly, given the n_i's, $V(X) = V_i$ with probability $n_i/(n+1)$. We need to show that $\mathbb{E}[V(X)] \to 0$. But we have

$$\mathbb{E}[V(X)] = \mathbb{E}\left[\frac{\sum_{i=1}^{k+1} n_i V_i}{n+1}\right].$$

So, it suffices to show that $\mathbb{E}[\sum_i n_i V_i] = o(n)$.

It is worthy of mention that the random split of a box can be imagined as follows. Given that we split along the s-th coordinate axis, and that a box has m points, then we select one of the $m+1$ spacings defined by these m points uniformly at random, still for that s-th coordinate. We cut that spacing properly but are free to do so anywhere. We can cut in proportions $\lambda, 1-\lambda$ with $\lambda \in (0,1)$, and the value of λ may vary from cut to cut and even be data-dependent. In fact, then, each internal and external node of our partition tree has associated with it two important quantities, a cardinality, and its first dimension. If we keep using i to index cells, then we can use n_i and V_i for the i-th cell, even if it is an internal cell.

Let A be the collection of external nodes in the subtree of the i-th cell. Then trivially,

$$\sum_{j \in A} n_j V_j \le n_i V_i \le n.$$

Thus, if E is the collection of all external nodes of a partition tree, ℓ is at most the minimum path distance from any cell in E to the root, and L is the collection of all nodes at distance ℓ from the root, then, by the last inequality,

$$\sum_{i \in E} n_i V_i \le \sum_{i \in L} n_i V_i.$$

Thus, using the notion of fill-up level F_n of the binary search tree, and setting $\ell = \lfloor 0.37 \log k \rfloor$, we have

$$\mathbb{E}\left[\sum_{i \in E} n_i V_i\right] \le n\mathbb{P}\{F_n < \ell\} + \mathbb{E}\left[\sum_{i \in L} n_i V_i\right].$$

We have seen that the first term is $o(n)$. We argue that the second term is not more than $n(1-1/(8d))^{\ell}$, which is $o(n)$ since $k \to \infty$. That will conclude the proof.

It suffices now to argue recursively and fix one cell of cardinality n and first dimension V. Let C be the collection of its children. We will show that

$$\mathbb{E}\left[\sum_{i\in C} n_i V_i\right] \le \left(1-\frac{1}{8d}\right) nV.$$

Repeating this recursively ℓ times shows that

$$\mathbb{E}\left[\sum_{i\in L} n_i V_i\right] \le n\left(1-\frac{1}{8d}\right)^{\ell}$$

because $V = 1$ at the root.

Fix that cell of cardinality n, and assume without loss of generality that $V = 1$. Let the spacings along the first coordinate be $a_1, \ldots, a_{n+1}$, their sum being one. With probability $1-1/d$, there the first axis is not cut, and thus, $\sum_{i\in C} n_i V_i = n$. With probability $1/d$, the first axis is cut in two parts. We will show that conditional on the event that the first direction is cut,

$$\mathbb{E}\left[\sum_{i} n_i V_i\right] \le \frac{7n}{8}\ .$$

Unconditionally, we have

$$\mathbb{E}\left[\sum_{i} n_i V_i\right] \le \left(1-\frac{1}{d}\right) n + \frac{1}{d}\cdot\frac{7n}{8} = \left(1-\frac{1}{8d}\right) n\ ,$$

as required. So, let us prove the conditional result.

Using δ_j to denote numbers drawn from $(0,1)$, possibly random, we have

$$\begin{aligned}
&\mathbb{E}\left[\sum_i n_i V_i\right] \\
&= \frac{1}{n+1}\mathbb{E}\Big[\sum_{j=1}^{n+1}\big[(j-1)(a_1+\cdots+a_{j-1}+a_j\delta_j) \\
&\qquad +(n+1-j)(a_j(1-\delta_j)+a_{j+1}+\cdots+a_{n+1})\big]\Big] \\
&= \frac{1}{n+1}\mathbb{E}\left[\sum_{k=1}^{n+1} a_k\left(\sum_{k<j\le n+1}(j-1)\right.\right. \\
&\qquad \left.\left.+\sum_{1\le j<k}(n+1-j)+\delta_k(k-1)+(1-\delta_k)(n+1-k)\right)\right] \\
&\le \frac{1}{n+1}\left(\sum_{k=1}^{n+1} a_k\left(n(n+1)-\frac{k(k-1)}{2}\right.\right. \\
&\qquad \left.\left.-\frac{(n-k+1)(n-k+2)}{2}+\max(k-1,n+1-k)\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{n+1}\left(\sum_{k=1}^{n+1} a_k \left(\frac{n(n+1)}{2} + (k-1)(n+1-k) + \max(k-1, n+1-k)\right)\right) \\
&\leq \frac{1}{n+1}\left(\left(\frac{n(n+1)}{2} + \left(\frac{n}{2}\right)^2 + n\right)\sum_{k=1}^{n+1} a_k\right) \\
&= n\left(\frac{3n/4 + (3/2)}{n+1}\right) \\
&\leq \frac{7n}{8} \quad \text{if } n > 4.
\end{aligned}
$$

□

Our definition of the scale-invariant random forest classifier permits cells to be cut such that one of the created cells becomes empty. One may easily prevent this by artificially forcing a minimum number of points in each cell. This may be done by restricting the random position of each cut so that both created subcells contain at least, say, m points. By a minor modification of the proof above it is easy to see that as long as m is bounded by a constant, the resulting random forest classifier remains consistent under the same conditions as in Theorem 4.

4. Creating Consistent Rules by Randomization

Proposition 1 shows that if a randomized classifier is consistent, then the corresponding averaged classifier remains consistent. The converse is not true. There exist inconsistent randomized classifiers such that their averaged version becomes consistent. Indeed, Breiman's (2001) original random forest classifier builds tree classifiers by successive randomized cuts until the cell of the point X to be classified contains only one data point, and classifies X as the label of this data point. Breiman's random forest classifier is just the averaged version of such randomized tree classifiers. The randomized base classifier $g_n(x,Z)$ is obviously not consistent for all distributions.

This does not imply that the averaged random forest classifier is not consistent. In fact, in this section we will see that averaging may "boost" inconsistent base classifiers into consistent ones. We point out in Section 6 that there are distributions of (X,Y) for which Breiman's random forest classifier is not consistent. The counterexample shown in Proposition 8 is such that the distribution of X doesn't have a density. It is possible, however, that Breiman's random forest classifier is consistent whenever the distribution of X has a density. Breiman's rule is difficult to analyze as each cut of the random tree is determined by a complicated function of the entire data set D_n (i.e., both feature vectors and labels). However, in Section 6 below we provide arguments suggesting that Breiman's random forest is not consistent when a density exists. Instead of Breiman's rule, next we analyze a stylized version by showing that inconsistent randomized rules that take the label of only one neighbor into account can be made consistent by averaging.

For simplicity, we consider the case $d = 1$, though the whole argument extends, in a straightforward way, to the multivariate case. To avoid complications introduced by ties, assume that X has a non-atomic distribution. Define a randomized nearest neighbor rule as follows: for a fixed $x \in \mathbb{R}$, let $X_{(1)}(x), X_{(2)}(x), \ldots, X_{(n)}(x)$ be the ordering of the data points $X_1, \ldots, X_n$ according to increasing distances from x. Let $U_1, \ldots, U_n$ be i.i.d. random variables, uniformly distributed over $[0,1]$. The vector of these random variables constitutes the randomization Z of the classifier. We define $g_n(x,Z)$

to be equal to the label $Y_{(i)}(x)$ of the data point $X_{(i)}(x)$ for which

$$\max(i, mU_i) \leq \max(j, mU_j) \quad \text{for all } j = 1, \ldots, n$$

where $m \leq n$ is a parameter of the rule. We call $X_{(i)}(x)$ the perturbed nearest neighbor of x. Note that $X_{(1)}(x)$ is the (unperturbed) nearest neighbor of x. To obtain the perturbed version, we artificially add a random uniform coordinate and select a data point with the randomized rule defined above. Since ties occur with probability zero, the perturbed nearest neighbor classifier is well defined almost surely. It is clearly not, in general, a consistent classifier.

Call the corresponding averaged classifier $\bar{g}_n(x) = \mathbb{1}_{\{\mathbb{E}_Z g_n(x,Z) \geq 1/2\}}$ the *averaged perturbed nearest neighbor classifier.*

In the proof of the consistency result below, we use Stone's (1977) general consistency theorem for locally weighted average classifiers, see also (Devroye, Györfi, and Lugosi, 1996, Theorem 6.3). Stone's theorem concerns classifiers that take the form

$$g_n(x) = \mathbb{1}_{\{\sum_{i=1}^n Y_i W_{ni}(x) \geq \sum_{i=1}^n (1-Y_i) W_{ni}(x)\}}$$

where the weights $W_{ni}(x) = W_{ni}(x, X_1, \ldots, X_n)$ are non-negative and sum to one. According to Stone's theorem, consistency holds if the following three conditions are satisfied:

(i)
$$\lim_{n\to\infty} \mathbb{E}\left[\max_{1\leq i\leq n} W_{ni}(X)\right] = 0.$$

(ii) For all $a > 0$,
$$\lim_{n\to\infty} \mathbb{E}\left[\sum_{i=1}^n W_{ni}(X)\, \mathbb{1}_{\{\|X_i - X\| > a\}}\right] = 0.$$

(iii) There is a constant c such that, for every non-negative measurable function f satisfying $\mathbb{E}f(X) < \infty$,
$$\mathbb{E}\left[\sum_{i=1}^n W_{ni}(X) f(X_i)\right] \leq c\mathbb{E}f(X).$$

Theorem 5 *The averaged perturbed nearest neighbor classifier $\bar{g}_n$ is consistent whenever the parameter m is such that $m \to \infty$ and $m/n \to 0$.*

Proof If we define

$$W_{ni}(x) = \mathbb{P}_Z\{X_i \text{ is the perturbed nearest neighbor of } x\}$$

then it is clear that the averaged perturbed nearest neighbor classifier is a locally weighted average classifier and Stone's theorem may be applied. It is convenient to introduce the notation

$$p_{ni}(x) = \mathbb{P}_Z\{X_{(i)}(x) \text{ is the perturbed nearest neighbor of } x\}$$

and write $W_{ni}(x) = \sum_{j=1}^n \mathbb{1}_{\{X_i = X_{(j)}(x)\}} p_{nj}(x)$.

To check the conditions of Stone's theorem, first note that

$$\begin{aligned} p_{ni}(x) &= \mathbb{P}\{mU_i \le i \le \min_{j<i} mU_j\} + \mathbb{P}\{i < mU_i \le \min_{j\le n} \max(j, mU_j)\} \\ &= \mathbb{1}_{\{i\le m\}} \frac{i}{m}\left(1-\frac{i}{m}\right)^{i-1} + \mathbb{P}\{i < mU_i \le \min_{j\le n} \max(j, mU_j)\}\ . \end{aligned}$$

Now we are prepared to check the conditions of Stone's theorem. To prove that (i) holds, note that by monotonicity of $p_{ni}(x)$ in i, it suffices to show that $p_{n1}(x) \to 0$.

But clearly, for $m \ge 2$,

$$\begin{aligned} p_{n1}(x) &\le \frac{1}{m} + \mathbb{P}\left\{U_1 \le \min_{j\le m} \max\left(\frac{j}{m}, U_j\right)\right\} \\ &= \frac{1}{m} + \mathbb{E}\left[\prod_{j=2}^{m} \mathbb{P}\left\{U_1 \le \max\left(\frac{j}{m}, U_j\right) | U_1\right\}\right] \\ &= \frac{1}{m} + \mathbb{E}\left[\prod_{j=2}^{m} \left[1 - \mathbb{1}_{\{U_1 > j/m\}} U_1\right]\right] \\ &\le \frac{1}{m} + \mathbb{E}\left[(1-U_1)^{mU_1-2} \mathbb{1}_{\{\lfloor mU_1\rfloor \ge 3\}}\right] + \mathbb{P}\left\{\lfloor mU_1\rfloor < 3\right\} \end{aligned}$$

which converges to zero by monotone convergence as $m \to \infty$.

(ii) follows by the condition $m/n \to 0$ since $\sum_{i=1}^n W_{ni}(X)\mathbb{1}_{\{\|X_i - X\| > a\}} = 0$ whenever the distance of m-th nearest neighbor of X to X is at most a. But this happens eventually, almost surely, see (Devroye, Györfi, and Lugosi, 1996, Lemma 5.1).

Finally, to check (iii), we use again the monotonicity of $p_{ni}(x)$ in i. We may write $p_{ni}(x) = a_i + a_{i+1} + \cdots + a_n$ for some non-negative numbers $a_j, 1 \le j \le n$, depending upon m and n but not x. Observe that $\sum_{j=1}^n j a_j = \sum_{i=1}^n p_{ni}(x) = 1$. But then

$$\begin{aligned} &\mathbb{E}\left[\sum_{i=1}^n W_{ni}(X) f(X_i)\right] \\ &= \mathbb{E}\left[\sum_{i=1}^n p_{ni}(X) f(X_{(i)})\right] \\ &= \mathbb{E}\left[\sum_{i=1}^n \sum_{j=i}^n a_j f(X_{(i)})\right] \\ &= \mathbb{E}\left[\sum_{j=1}^n a_j \sum_{i=1}^j f(X_{(i)})\right] \\ &= \sum_{j=1}^n a_j \mathbb{E}\left[\sum_{i=1}^j f(X_{(i)})\right] \end{aligned}$$

$$\leq \quad c\sum_{j=1}^{n} a_j j\mathbb{E}f(X)$$

(by Stone's (1977) lemma, see (Devroye, Györfi, and Lugosi, 1996, Lemma 5.3), where c is a constant)

$$= \quad c\mathbb{E}f(X)\sum_{j=1}^{n} a_j j = c\mathbb{E}f(X)$$

as desired. □

5. Bagging

One of the first and simplest ways of randomizing and averaging classifiers in order to improve their performance is *bagging*, suggested by Breiman (1996). In bagging, randomization is achieved by generating many bootstrap samples from the original data set. Breiman suggests selecting n training pairs (X_i,Y_i) at random, *with replacement* from the *bag* of all training pairs $\{(X_1,Y_1),\ldots,(X_n,Y_n)\}$. Denoting the random selection process by Z, this way one obtains new training data $D_n(Z)$ with possible repetitions and given a classifier $g_n(X,D_n)$, one can calculate the randomized classifier $g_n(X,Z,D_n) = g_n(X,D_n(Z))$. Breiman suggests repeating this procedure for many independent draws of the bootstrap sample, say m of them, and calculating the voting classifier $g_n^{(m)}(X,Z^m,D_n)$ as defined in Section 2.

In this section we consider a generalized version of bagging predictors in which the size of the bootstrap samples is not necessary the same as that the original sample. Also, to avoid complications and ambiguities due to replicated data points, we exclude repetitions in the bootstrapped data. This is assumed for convenience but sampling with replacement can be treated by minor modifications of the arguments below.

To describe the model we consider, introduce a parameter $q_n \in [0,1]$. In the bootstrap sample $D_n(Z)$ each data pair (X_i,Y_i) is present with probability q_n, independently of each other. Thus, the size of the bootstrapped data is a binomial random variable N with parameters n and q_n. Given a sequence of (non-randomized) classifiers $\{g_n\}$, we may thus define the randomized classifier

$$g_n(X,Z,D_n) = g_N(X,D_n(Z)) \ ,$$

that is, the classifier is defined based on the randomly re-sampled data. By drawing m independent bootstrap samples $D_n(Z_1),\ldots,D_n(Z_m)$ (with sizes $N_1,\ldots,N_m$), we may define the *bagging classifier* $g_n^{(m)}(X,Z^m,D_n)$ as the voting classifier based on the randomized classifiers $g_{N_1}(X,D_n(Z_1)),\ldots,$ $g_{N_m}(X,D_n(Z_m))$ as in Section 2. For the theoretical analysis it is more convenient to consider the averaged classifier $\bar{g}_n(x,D_n) = \mathbb{1}_{\{\mathbb{E}_Z g_N(x,D_n(Z))\geq 1/2\}}$ which is the limiting classifier one obtains as the number m of the bootstrap replicates grows to infinity.

The following result establishes consistency of bagging classifiers under the assumption that the original classifier is consistent. It suffices that the expected size of the bootstrap sample goes to infinity. The result is an immediate consequence of Proposition 1. Note that the choice of m does not matter in Theorem 6. It can be one, constant, or a function of n.

Theorem 6 *Let $\{g_n\}$ be a sequence of classifiers that is consistent for the distribution of (X,Y). Consider the bagging classifiers $g_n^{(m)}(x,Z^m,D_n)$ and $\bar{g}_n(x,D_n)$ defined above, using parameter q_n. If $nq_n \to \infty$ as $n \to \infty$ then both classifiers are consistent.*

If a classifier is insensitive to duplicates in the data, Breiman's original suggestion is roughly equivalent to taking $q_n \approx 1-1/e$.

However, it may be advantageous to choose much smaller values of q_n. In fact, small values of q_n may turn inconsistent classifiers into consistent ones via the bagging procedure. We illustrate this phenomenon on the simple example of the 1-nearest neighbor rule.

Recall that the 1-nearest neighbor rule sets $g_n(x,D_n) = Y_{(1)}(x)$ where $Y_{(1)}(x)$ is the label of the feature vector $X_{(1)}(x)$ whose Euclidean distance to x is minimal among all $X_1,\ldots,X_n$. Ties are broken in favor of smallest indices. It is well-known that g_n is consistent only if either $L^* = 0$ or $L^* = 1/2$, otherwise its asymptotic probability of error is strictly greater than L^*. However, by bagging one may turn the 1-nearest neighbor classifier into a consistent one, provided that the size of the bootstrap sample is sufficiently small. The next result characterizes consistency of the bagging version of the 1-nearest neighbor classifier in terms of the parameter q_n.

Theorem 7 *The bagging averaged* 1*-nearest neighbor classifier* $\overline{g}_n(x,D_n)$ *is consistent for all distributions of* (X,Y) *if and only if* $q_n \to 0$ *and* $nq_n \to \infty$.

Proof It is obvious that both $q_n \to 0$ and $nq_n \to \infty$ are necessary for consistency for all distributions.

Assume now that $q_n \to 0$ and $nq_n \to \infty$. The key observation is that $\overline{g}_n(x,D_n)$ is a locally weighted average classifier for which Stone's consistency theorem, recalled in Section 4, applies.

Recall that for a fixed $x \in \mathbb{R}$, $X_{(1)}(x), X_{(2)}(x), \ldots, X_{(n)}(x)$ denotes the ordering of the data points $X_1,\ldots,X_n$ according to increasing distances from x. (Points with equal distances to x are ordered according to their indices.) Observe that $\overline{g}_n$ may be written as

$$\overline{g}_n(x,D_n) = \mathbb{1}_{\{\sum_{i=1}^n Y_i W_{ni}(x) \geq \sum_{i=1}^n (1-Y_i) W_{ni}(x)\}}$$

where $W_{ni}(x) = \sum_{j=1}^n \mathbb{1}_{\{X_i = X_{(j)}(x)\}} p_{nj}(x)$ and $p_{ni}(x) = (1-q_n)^{i-1} q_n$ is defined as the probability (with respect to the random selection Z of the bootstrap sample) that $X_{(i)}(x)$ is the nearest neighbor of x in the sample $D_n(Z)$. It suffices to prove that the weights $W_{ni}(X)$ satisfy the three conditions of Stone's theorem.

Condition (i) obviously holds because $\max_{1\leq i\leq n} W_{ni}(X) = p_{n1}(X) = q_n \to 0$.

To check condition (ii), define $k_n = \left\lceil \sqrt{n/q_n} \right\rceil$. Since $nq_n \to \infty$ implies that $k_n/n \to 0$, it follows from (Devroye, Györfi, and Lugosi, 1996, Lemma 5.1) that eventually, almost surely, $\|X - X_{(k_n)}(X)\| \leq a$ and therefore

$$\begin{aligned}
\sum_{i=1}^n W_{ni}(X) \mathbb{1}_{\{\|X_i - X\| > a\}} &\leq \sum_{i=k_n+1}^n p_{ni}(X) \\
&= \sum_{i=k_n+1}^n q_n (1-q_n)^{i-1} \\
&\leq (1-q_n)^{k_n} \\
&\leq (1-q_n)^{\sqrt{n/q_n}} \\
&\leq e^{-\sqrt{nq_n}}
\end{aligned}$$

where we used $1 - q_n \leq e^{-q_n}$. Therefore, $\sum_{i=1}^n W_{ni}(X) \mathbb{1}_{\{\|X_i - X\| > a\}} \to 0$ almost surely and Stone's second condition is satisfied by dominated convergence.

Finally, condition (iii) follows from the fact that $p_{ni}(x)$ is monotone decreasing in i, after using an argument as in the proof of Theorem 5. $\square$

6. Random Forests Based on Greedily Grown Trees

In this section we study random forest classifiers that are based on randomized tree classifiers that are constructed in a greedy manner, by recursively splitting cells to minimize an empirical error criterion. Such greedy forests were introduced by Breiman (2001, 2004) and have shown excellent performance in many applications. One of his most popular classifiers is an averaging classifier, $\overline{g}_n$, based on a randomized tree classifier $g_n(x,Z)$ defined as follows. The algorithm has a parameter $1 \leq v < d$ which is a positive integer. The feature space $\mathbb{R}^d$ is partitioned recursively to form a tree partition. The root of the random tree is $\mathbb{R}^d$. At each step of the construction of the tree, a leaf is chosen uniformly at random. v variables are selected uniformly at random from the d candidates $x^{(1)},\ldots,x^{(d)}$. A split is selected along one of these v variables to minimize the number of misclassified training points if a majority vote is used in each cell. The procedure is repeated until every cell contains exactly one training point X_i. (This is always possible if the distribution of X has non-atomic marginals.)

In some versions of Breiman's algorithm, a bootstrap subsample of the training data is selected before the construction of each tree to increase the effect of randomization.

As observed by Lin and Jeon (2006), Breiman's classifier is a weighted *layered nearest neighbor* classifier, that is, a classifier that takes a (weighted) majority vote among the layered nearest neighbors of the observation x. X_i is called a layered nearest neighbor of x if the rectangle defined by x and X_i as their opposing vertices does not contain any other data point X_j ($j \neq i$). This property of Breiman's random forest classifier is a simple consequence of the fact that each tree is grown until every cell contains just one data point. Unfortunately, this simple property prevents the random tree classifier from being consistent for all distributions:

Proposition 8 *There exists a distribution of (X,Y) such that X has non-atomic marginals for which Breiman's random forest classifier is not consistent.*

Proof The proof works for any weighted layered nearest neighbor classifier. Let the distribution of X be uniform on the segment $\{x = (x^{(1)},\ldots,x^{(d)}) : x^{(1)} = \cdots = x^{(d)}, x^{(1)} \in [0,1]\}$ and let the distribution of Y be such that $L^* \neq \{0,1/2\}$. Then with probability one, X has only two layered nearest neighbors and the classification rule is not consistent. (Note that Problem 11.6 in Devroye, Györfi, and Lugosi 1996 erroneously asks the reader to prove consistency of the (unweighted) layered nearest neighbor rule for any distribution with non-atomic marginals. As the example in this proof shows, the statement of the exercise is incorrect. Consistency of the layered nearest neighbor rule is true however, if the distribution of X has a density.) $\square$

One may also wonder whether Breiman's random forest classifier is consistent if instead of growing the tree down to cells with a single data point, one uses a different stopping rule, for example if one fixes the total number of cuts at k and let k grow slowly as in the examples of Section 3. The next two-dimensional example provides an indication that this is not necessarily the case. Consider the joint distribution of (X,Y) sketched in Figure 1. X has a uniform distribution on $[0,1]\times[0,1] \cup [1,2]\times[1,2] \cup [2,3]\times[2,3]$. Y is a function of X, that is $\eta(x) \in \{0,1\}$ and $L^* = 0$. The lower left square $[0,1]\times[0,1]$ is divided into countably infinitely many vertical stripes in

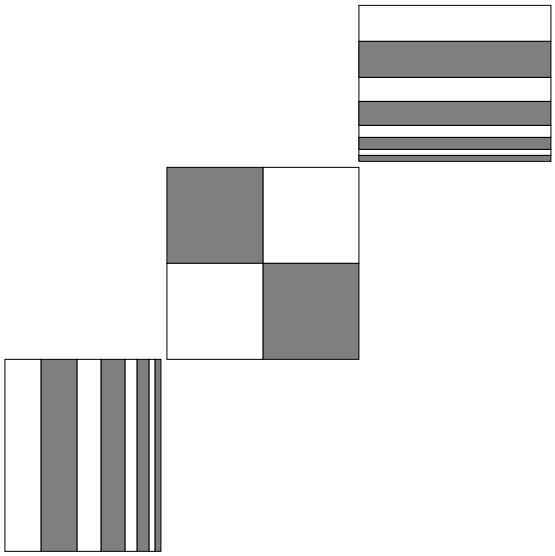


Figure 1: An example of a distribution for which greedy random forests are inconsistent. The distribution of X is uniform on the union of the three large squares. White areas represent the set where $\eta(x) = 0$ and on the grey regions $\eta(x) = 1$.

which the stripes with $\eta(x) = 0$ and $\eta(x) = 1$ alternate. The upper right square $[2,3] \times [2,3]$ is divided similarly into horizontal stripes. The middle rectangle $[1,2] \times [1,2]$ is a 2×2 checkerboard. Consider Breiman's random forest classifier with $v = 1$ (the only possible choice when $d = 2$).

For simplicity, consider the case when, instead of minimizing the empirical error, each tree is grown by minimizing the true probability of error at each split in each random tree. Then it is easy to see that no matter what the sequence of random selection of split directions is and no matter for how long each tree is grown, no tree will ever cut the middle rectangle and therefore the probability of error of the corresponding random forest classifier is at least $1/6$.

It is not so clear what happens in this example if the successive cuts are made by minimizing the empirical error. Whether the middle square is ever cut will depend on the precise form of the stopping rule and the exact parameters of the distribution. The example is here to illustrate that consistency of greedily grown random forests is a delicate issue. Note however that if Breiman's original algorithm is used in this example (i.e., when all cells with more than one data point in it are split) then one obtains a consistent classification rule. If, on the other hand, horizontal or vertical cuts are selected to minimize the probability of error, and $k \to \infty$ in such a way that $k = O(n^{1/2-\varepsilon})$ for some $\varepsilon > 0$, then, as errors on the middle square are never more than about $O(1/\sqrt{n})$ (by the limit law for the Kolmogorov-Smirnov statistic), we see that thin strips of probability mass more than $1/\sqrt{n}$ are preferentially cut. By choosing the probability weights of the strips, one can easily see that we can construct more than $2k$ such strips. Thus, when $k = O(n^{1/2-\varepsilon})$, no consistency is possible on that example.

We note here that many versions of random forest classifiers build on random tree classifiers based on bootstrap subsampling. This is the case of Breiman's principal random forest classifier.

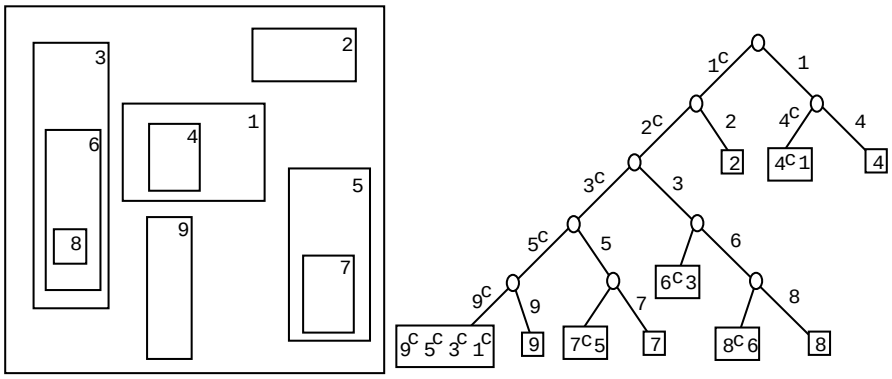


Figure 2: A tree based on partitioning the plane into rectangles. The right subtree of each internal node belongs to the inside of a rectangle, and the left subtree belongs to the complement of the same rectangle (i^c denotes the complement of i). Rectangles are not allowed to overlap.

Breiman suggests to take a random sample of size n drawn with replacement from the original data. While this may result in an improved behavior in some practical instances, it is easy to see that such a subsampling procedure does not vary the consistency property of any of the classifiers studied in this paper. For example, non-consistency of Breiman's random forest classifier with bootstrap resampling for the distribution considered in the proof of Proposition 8 follows from the fact that the two layered nearest neighbors on both sides are included in the bootstrap sample with a probability bounded away from zero and therefore the weight of these two points is too large, making consistency impossible.

In order to remedy the inconsistency of greedily grown tree classifiers, (Devroye, Györfi, and Lugosi, 1996, Section 20.14) introduce a greedy tree classifier which, instead of cutting every cell along just one direction, cuts out a whole hyper-rectangle from a cell in a way to optimize the empirical error. The disadvantage of this method is that in each step, d parameters need to be optimized jointly and this may be computationally prohibitive if d is not very small. (The computational complexity of the method is $O(n^d)$.) However, we may use the methodology of random forests to define a computationally feasible consistent greedily grown random forest classifier.

In order to define the consistent greedy random forest, we first recall the tree classifier of (Devroye, Györfi, and Lugosi, 1996, Section 20.14).

The space is partitioned into rectangles as shown in Figure 2.

A hyper-rectangle defines a split in a natural way. A partition is denoted by $\mathcal{P}$, and a decision on a set $A \in \mathcal{P}$ is by majority vote. We write $g_{\mathcal{P}}$ for such a rule:

$$g_{\mathcal{P}}(x) = \mathbb{1}_{\{\sum_{i:X_i \in A(x)} Y_i > \sum_{i:X_i \in A(x)} (1-Y_i)\}}$$

where $A(x)$ denotes the cell of the partition containing x. Given a partition $\mathcal{P}$, a legal hyper-rectangle T is one for which $T \cap A = \emptyset$ or $T \subseteq A$ for all sets $A \in \mathcal{P}$. If we refine $\mathcal{P}$ by adding a legal rectangle T somewhere, then we obtain the partition $\mathcal{T}$. The decision $g_{\mathcal{T}}$ agrees with $g_{\mathcal{P}}$ except on the set $A \in \mathcal{P}$ that contains T.

Introduce the convenient notation

$$\begin{aligned} \nu_j(A) &= \mathbb{P}\{X \in A, Y = j\},\ j \in \{0,1\}, \\ \nu_{j,n}(A) &= \frac{1}{n}\sum_{i=1}^{n} I_{\{X_i \in A, Y_i = j\}},\ j \in \{0,1\}. \end{aligned}$$

The empirical error of $g_{\mathcal{P}}$ is

$$\widehat{L}_n(\mathcal{P}) \stackrel{\text{def}}{=} \sum_{R \in \mathcal{P}} \widehat{L}_n(R),$$

where

$$\widehat{L}_n(R) = \frac{1}{n}\sum_{i=1}^{n} I_{\{X_i \in R, g_{\mathcal{P}}(X_i) \neq Y_i\}} = \min(\nu_{0,n}(R), \nu_{1,n}(R)).$$

We may similarly define $\widehat{L}_n(\mathcal{T})$. Given a partition $\mathcal{P}$, the greedy classifier selects that legal rectangle T for which $\widehat{L}_n(\mathcal{T})$ is minimal (with any appropriate policy for breaking ties). Let R be the set of $\mathcal{P}$ containing T. Then the greedy classifier picks that T for which

$$\widehat{L}_n(T) + \widehat{L}_n(R - T) - \widehat{L}_n(R)$$

is minimal. Starting with the trivial partition $\mathcal{P}_0 = \{\mathbb{R}^d\}$, we repeat the previous step k times, leading thus to $k+1$ regions. The sequence of partitions is denoted by $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_k$.

(Devroye, Györfi, and Lugosi, 1996, Theorem 20.9) establish consistency of this classifier. More precisely, it is shown that if X has non-atomic marginals, then the greedy classifier with $k \to \infty$ and $k = o\left(\sqrt{n/\log n}\right)$ is consistent.

Based on the greedy tree classifier, we may define a random forest classifier by considering its bagging version. More precisely, let $q_n \in [0,1]$ be a parameter and let $Z = Z(D_n)$ denote a random subsample of size binomial (n, q_n) of the training data (i.e., each pair (X_i, Y_i) is selected at random, without replacement, from D_n, with probability q_n) and let $g_n(x, Z)$ be the greedy tree classifier (as defined above) based on the training data $Z(D_n)$. Define the corresponding averaged classifier $\overline{g}_n$. We call $\overline{g}_n$ the *greedy random forest* classifier. Note that $\overline{g}_n$ is just the bagging version of the greedy tree classifier and therefore Theorem 6 applies:

Theorem 9 *The greedy random forest classifier is consistent whenever X has non-atomic marginals in $\mathbb{R}^d$, $nq_n \to \infty$, $k \to \infty$ and $k = o\left(\sqrt{nq_n/\log(nq_n)}\right)$ as $n \to \infty$.*

Proof This follows from Theorem 6 and the fact that the greedy tree classifier is consistent (see Theorem 20.9 of Devroye, Györfi, and Lugosi (1996)). □

Observe that the computational complexity of building the randomized tree classifier $g_n(x, Z)$ is $O((nq_n)^d)$. Thus, the complexity of computing the voting classifier $g_n^{(m)}$ is $m(nq_n)^d$. If $q_n \ll 1$, this may be a significant speed-up compared to the complexity $O(n^d)$ of computing a single tree classifier using the full sample. Repeated subsampling and averaging may make up for the effect of decreased sample size.

Acknowledgments

We thank James Malley for stimulating discussions. We also thank three referees for valuable comments and insightful suggestions.

The second author's research was sponsored by NSERC Grant A3456 and FQRNT Grant 90-ER-0291. The third author acknowledges support by the Spanish Ministry of Science and Technology grant MTM2006-05650 and by the PASCAL Network of Excellence under EC grant no. 506778.

References

Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Arcing classifiers. *The Annals of Statistics*, 24:801–849, 1998.

L. Breiman. Some infinite theory for predictor ensembles. *Technical Report 577*, Statistics Department, UC Berkeley, 2000. http://www.stat.berkeley.edu/˜breiman .

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

L. Breiman. Consistency for a simple model of random forests. *Technical Report 670*, Statistics Department, UC Berkeley, 2004.

A. Cutler and G. Zhao. Pert – Perfect random tree ensembles, *Computing Science and Statistics*, 33:490–497, 2001.

L. Devroye. Applications of the theory of records in the study of random trees. *Acta Informatica*, 26:123–130, 1988.

L. Devroye. A note on the height of binary search trees. *Journal of the ACM*, 33:489–498, 1986.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, New York, 1996.

T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40:139–157, 2000.

T.G. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli (Eds.), *First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, pp. 1–15, Springer-Verlag, New York, 2000.

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In L. Saitta (Ed.), *Machine Learning: Proceedings of the 13th International Conference*, pp. 148–156, Morgan Kaufmann, San Francisco, 1996.

Y. Lin and Y. Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101:578–590, 2006.

N. Meinshausen. Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

H.M. Mahmoud. *Evolution of Random Search Trees*. John Wiley, New York, 1992.

C. Stone. Consistent nonparametric regression. *The Annals of Statistics*, 5:595–645, 1977.

Journal of Machine Learning Research 9 (2008) 2035-2078 Submitted 8/07; Revised 4/08; Published 10/08

Approximations for Binary Gaussian Process Classification

Hannes Nickisch HN@TUEBINGEN.MPG.DE
Max Planck Institute for Biological Cybernetics
Spemannstraße 38
72076 Tübingen, Germany

Carl Edward Rasmussen* CER54@CAM.AC.UK
Department of Engineering
University of Cambridge
Trumpington Street
Cambridge, CB2 1PZ, UK

Editor: Carlos Guestrin

Abstract

We provide a comprehensive overview of many recent algorithms for approximate inference in Gaussian process models for probabilistic binary classification. The relationships between several approaches are elucidated theoretically, and the properties of the different algorithms are corroborated by experimental results. We examine both 1) the quality of the predictive distributions and 2) the suitability of the different marginal likelihood approximations for model selection (selecting hyperparameters) and compare to a gold standard based on MCMC. Interestingly, some methods produce good predictive distributions although their marginal likelihood approximations are poor. Strong conclusions are drawn about the methods: The Expectation Propagation algorithm is almost always the method of choice unless the computational budget is very tight. We also extend existing methods in various ways, and provide unifying code implementing all approaches.

Keywords: Gaussian process priors, probabilistic classification, Laplaces's approximation, expectation propagation, variational bounding, mean field methods, marginal likelihood evidence, MCMC

1. Introduction

Gaussian processes (GPs) can conveniently be used to specify prior distributions for Bayesian inference. In the case of regression with Gaussian noise, inference can be done simply in closed form, since the posterior is also a GP. For non-Gaussian likelihoods, such as e.g., in binary classification, exact inference is analytically intractable.

One prolific line of attack is based on approximating the non-Gaussian posterior with a tractable Gaussian distribution. One might think that finding such an approximating GP is a well-defined problem with a largely unique solution. However, we find no less than three different types of solution in the recent literature: Laplace Approximation (LA) (Williams and Barber, 1998), Expectation Propagation (EP) (Minka, 2001a) and Kullback-Leibler divergence (KL) minimization (Opper and Archambeau, 2008) comprising Variational Bounding (VB) (Gibbs and MacKay, 2000) as a special

∗. Also at Max Planck Institute for Biological Cybernetics, Spemannstraße 38, 72076 Tübingen, Germany.

case. Another approach is based on a factorial approximation, rather than a Gaussian (Csató et al., 2000).

Practical applications reflect the richness of approximate inference methods: LA has been used for sequence annotation (Altun et al., 2004) and prostate cancer prediction (Chu et al., 2005), EP for affect recognition (Kapoor and Picard, 2005), VB for weld cracking prognosis (Gibbs and MacKay, 2000), Label Regression (LR) serves for object categorization (Kapoor et al., 2007) and MCMC sampling is applied to rheuma diagnosis (Schwaighofer et al., 2002). Brain computer interfaces (Zhong et al., 2008) even rely on several (LA, EP, VB) methods.

In this paper, we compare these different approximations and provide insights into the strengths and weaknesses of each method, extending the work of Kuss and Rasmussen (2005) in several directions: We cover many more approximation methods (VB,KL,FV,LR), put all of them in common framework and provide generic implementations dealing with both the logistic and the cumulative Gaussian likelihood functions and clarify the aspects of the problem causing difficulties for each method. We derive Newton's method for KL and VB. We show how to accelerate MCMC simulations. We highlight numerical problems, comment on computational complexity and supply runtime measurements based on experiments under a wide range of conditions, including different likelihood and different covariance functions. We provide deeper insights into the methods behavior by systematically linking them to each other. Finally, we review the tight connections to methods from the literature on Statistical Physics, including the TAP approximation and TAPnaive.

The quantities of central importance are the quality of the probabilistic predictions and the suitability of the approximate marginal likelihood for selecting parameters of the covariance function (hyperparameters). The marginal likelihood for any Gaussian approximate posterior can be lower bounded using Jensen's inequality, but the specific approximation schemes also come with their own marginal likelihood approximations.

We are able to draw clear conclusions. Whereas every method has good performance under some circumstances, only a single method gives consistently good results. We are able to theoretically corroborate our experimental findings; together this provides solid evidence and guidelines for choosing an approximation method in practice.

2. Gaussian Processes for Binary Classification

We describe probabilistic binary classification based on Gaussian processes in this section. For a graphical model representation see Figure 1 and for a 1d pictorial description consult Figure 2. Given data points $\mathbf{x}_i$ from a domain $\mathcal{X}$ with corresponding class labels $y_i \in \{-1,+1\}$, one would like to predict the class membership probability for a test point $\mathbf{x}_*$. This is achieved by using a *latent function* f whose value is mapped into the unit interval by means of a sigmoid function $\mathrm{sig}:\mathbb{R}\to[0,1]$ such that the class membership probability $\mathbb{P}(y=+1|\mathbf{x})$ can be written as $\mathrm{sig}(f(\mathbf{x}))$. The class membership probability must normalize $\sum_y \mathbb{P}(y|\mathbf{x})=1$, which leads to $\mathbb{P}(y=+1|\mathbf{x})=1-\mathbb{P}(y=-1|\mathbf{x})$. If the sigmoid function satisfies the point symmetry condition $\mathrm{sig}(t)=1-\mathrm{sig}(-t)$, the *likelihood* can be compactly written as

$$\mathbb{P}(y|\mathbf{x}) \quad = \quad \mathrm{sig}(y\cdot f(\mathbf{x}))\,.$$

In this paper, two point symmetric sigmoids are considered

$$\begin{aligned} \mathrm{sig}_{\mathrm{logit}}(t) &:= \frac{1}{1+e^{-t}} \\ \mathrm{sig}_{\mathrm{probit}}(t) &:= \int_{-\infty}^{t} \mathcal{N}(\tau|0,1)\mathrm{d}\tau. \end{aligned}$$

The two functions are very similar at the origin (showing locally linear behavior around $\mathrm{sig}(0) = 1/2$ with slope $1/4$ for $\mathrm{sig}_{\mathrm{logit}}$ and $1/\sqrt{2\pi}$ for $\mathrm{sig}_{\mathrm{probit}}$) but differ in how fast they approach $0/1$ when t goes to infinity. For large negative values of t, we have the asymptotics

$$\mathrm{sig}_{\mathrm{logit}}(t) \approx \exp(-t) \quad \text{and} \quad \mathrm{sig}_{\mathrm{probit}}(t) \approx \exp(-\frac{1}{2}t^2 + 0.158t - 1.78), \text{ for } t \ll 0.$$

Linear decay of $\ln(\mathrm{sig}_{\mathrm{logit}})$ corresponds to a weaker penalty for wrongly classified examples than the quadratic decay of $\ln(\mathrm{sig}_{\mathrm{probit}})$.

For notational convenience, the following shorthands are used: The matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ of size $n \times d$ collects the training points, the vector $\mathbf{y} = [y_1, \ldots, y_n]^\top$ of size $n \times 1$ collects the target values and latent function values are summarized by $\mathbf{f} = [f_1, \ldots, f_n]^\top$ with $f_i = f(\mathbf{x}_i)$. Observed data is written as $\mathcal{D} = \{(\mathbf{x}_i, y_i) \,|\, i = 1, \ldots, n\} = (\mathbf{X}, \mathbf{y})$. Quantities carrying an asterisk refer to test points, that is, $\mathbf{f}_*$ contains the latent function values for test points $[\mathbf{x}_{*,1}, \ldots, \mathbf{x}_{*,m}] = \mathbf{X}_* \subset \mathcal{X}$. Covariances between latent values $\mathbf{f}$ and $\mathbf{f}_*$ at data points $\mathbf{x}$ and $\mathbf{x}_*$ follow the same notation, namely $[\mathbf{K}_{**}]_{ij} = k(\mathbf{x}_{*,i}, \mathbf{x}_{*,j})$, $[\mathbf{K}_*]_{ij} = k(\mathbf{x}_i, \mathbf{x}_{*,j})$, $[\mathbf{k}_*]_i = k(\mathbf{x}_i, \mathbf{x}_*)$ and $k_{**} = k(x_*, x_*)$, where $[\mathbf{A}]_{ij}$ denotes the entry A_{ij} of the matrix $\mathbf{A}$.

Given the latent function f, the class labels are assumed to be Bernoulli distributed and independent random variables, which gives rise to a *factorial likelihood*, factorizing over data points (see Figure 1)

$$\mathbb{P}(\mathbf{y}|f) \;=\; \mathbb{P}(\mathbf{y}|\mathbf{f}) \;=\; \prod_{i=1}^{n} \mathbb{P}(y_i|f_i) \;=\; \prod_{i=1}^{n} \mathrm{sig}(y_i f_i). \tag{1}$$

A GP (Rasmussen and Williams, 2006) is a stochastic process fully specified by a *mean function* $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and a positive definite *covariance function* $k(\mathbf{x}, \mathbf{x}') = \mathbb{V}[f(\mathbf{x}), f(\mathbf{x}')]$. This means that a random variable $f(\mathbf{x})$ is associated to every $\mathbf{x} \in \mathcal{X}$, such that for any set of inputs $\mathbf{X} \subset \mathcal{X}$, the joint distribution $\mathbb{P}(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_0, \mathbf{K})$ is Gaussian with mean vector $\mathbf{m}_0$ and covariance matrix $\mathbf{K}$. The mean function and covariance functions may depend on additional *hyperparameters* $\boldsymbol{\theta}$. For notational convenience we will assume $m(x) \equiv 0$ throughout. Thus, the elements of $\mathbf{K}$ are $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\theta})$.

By application of Bayes' rule, one gets an expression for the *posterior* distribution over the latent values $\mathbf{f}$

$$\mathbb{P}(\mathbf{f}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \;=\; \frac{\mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})}{\int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})\,\mathrm{d}\mathbf{f}} \;=\; \frac{\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})}{\mathbb{P}(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})} \prod_{i=1}^{n} \mathrm{sig}(y_i f_i), \tag{2}$$

where $Z = \mathbb{P}(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta})\,\mathrm{d}\mathbf{f}$ denotes the *marginal likelihood* or *evidence* for the hyperparameter $\boldsymbol{\theta}$. The joint prior over training and test latent values $\mathbf{f}$ and $\mathbf{f}_*$ given the corresponding inputs is

$$\mathbb{P}(\mathbf{f}_*, \mathbf{f} | \mathbf{X}_*, \mathbf{X}, \boldsymbol{\theta}) \quad = \quad \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{bmatrix} \right).$$

When making predictions, we marginalize over the training set latent variables

$$\mathbb{P}(\mathbf{f}_* | \mathbf{X}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) = \int \mathbb{P}(\mathbf{f}_*, \mathbf{f} | \mathbf{X}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{f} = \int \mathbb{P}(\mathbf{f}_* | \mathbf{f}, \mathbf{X}_*, \mathbf{X}, \boldsymbol{\theta}) \, \mathbb{P}(\mathbf{f} | \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{f}, \tag{3}$$

where the joint posterior is factored into the product of the posterior and the conditional prior

$$\mathbb{P}(\mathbf{f}_* | \mathbf{f}, \mathbf{X}_*, \mathbf{X}, \boldsymbol{\theta}) \quad = \quad \mathcal{N}\left(\mathbf{f}_* | \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{f}, \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}^{-1} \mathbf{K}_* \right).$$

Finally, the predictive class membership probability $p_* := \mathbb{P}(y_* = 1 | \mathbf{x}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta})$ is obtained by averaging out the test set latent variables

$$\mathbb{P}(y_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) = \int \mathbb{P}(y_* | f_*) \, \mathbb{P}(f_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \, d f_* \; = \; \int \mathrm{sig}(y_* f_*) \, \mathbb{P}(f_* | \mathbf{x}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \, \mathrm{d} f_*. \tag{4}$$

The integral is analytically tractable for $\mathrm{sig}_{\mathrm{probit}}$ (Rasmussen and Williams, 2006, Ch. 3.9) and can be efficiently approximated for $\mathrm{sig}_{\mathrm{logit}}$ (Williams and Barber, 1998, App. A).

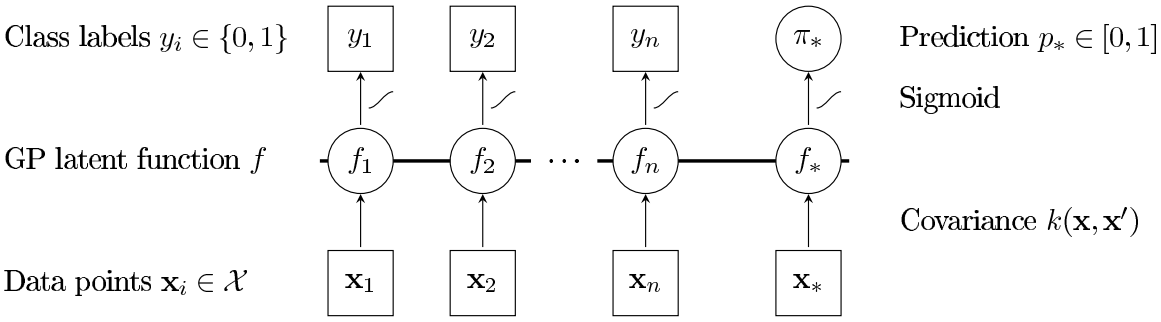


Figure 1: Graphical Model for binary Gaussian process classification: Circles represent unknown quantities, squares refer to observed variables. The horizontal thick line means fully connected latent variables. An observed label y_i is conditionally independent of all other nodes given the corresponding latent variable f_i. Labels y_i and latent function values f_i are connected through the sigmoid likelihood; all latent function values f_i are fully connected, since they are drawn from the same GP. The labels y_i are binary, whereas the prediction p_* is a probability and can thus have values from the whole interval $[0,1]$.

2.1 Stationary Covariance Functions

In preparation for the analysis of the approximation schemes described in this paper, we investigate some simple properties of the posterior for stationary covariance functions in different regimes

encountered in classification. Stationary covariances of the form $k(\mathbf{x},\mathbf{x}',\boldsymbol{\theta}) = \sigma_f^2 g(|\mathbf{x}-\mathbf{x}'|/\ell)$ with $g:\mathbb{R}\to\mathbb{R}$ a monotonously decreasing function[1] and $\boldsymbol{\theta} = \{\sigma_f,\ell\}$ are widely used. The following section supplies a geometric intuition of that specific prior in the classification scenario by analyzing the limiting behavior of the covariance matrix $\mathbf{K}$ as a function of the length scale ℓ and the limiting behavior of the likelihood as a function of the latent function scale σ_f. A pictorial illustration of the setting is given in Figure 3.

2.1.1 LENGTH SCALE

Two limiting cases of "ignorance with respect to the data" with marginal likelihood $Z = 2^{-n}$ can be distinguished, where $\mathbb{1} = [1,\dots 1]^\top$ and $\mathbf{I}$ is the identity matrix (see Appendix B.1)

$$\begin{aligned}\lim_{\ell\to 0}\mathbf{K} &= \sigma_f^2\mathbf{I},\\ \lim_{\ell\to\infty}\mathbf{K} &= \sigma_f^2\mathbb{1}\mathbb{1}^\top.\end{aligned}$$

For very small length scales ($\ell\to 0$), the prior is simply isotropic as all points are deemed to be far away from each other and the whole model factorizes. Thus, the (identical) posterior moments can be calculated dimension-wise. (See Figure 3, regimes 1, 4 and 7.)

For very long length scales ($\ell\to\infty$), the prior becomes degenerate as all datapoints are deemed to be close to each other and takes the form of a cigar along the hyper-diagonal. (See Figure 3, regimes 3, 6 and 9.) A 1d example of functions drawn from GP priors with different lengthscales ℓ is shown in Figure 2 on the left. The lengthscale has to be suited to the data; if chosen too small, we will overfit, if chosen too high underfitting will occur.

2.1.2 LATENT FUNCTION SCALE

The sigmoid likelihood function $\mathrm{sig}(y_i f_i)$ measures the agreement of the signs of the latent function and the label in a smooth way, that is, values close to one if the signs of y_i and f_i are the same and $|f_i|$ is large, and values close to zero if the signs are different and $|f_i|$ is large. The latent function scale σ_f of the data can be moved into the likelihood $\tilde{\mathrm{sig}}_{\sigma_f}(t) = \mathrm{sig}(\sigma_f^2 t)$, thus σ_f models the steepness of the likelihood and finally the smoothness of the agreement by interpolation between the two limiting cases "ignorant" and "hard cut"

$$\begin{aligned}\lim_{\sigma_f\to 0}\mathrm{sig}(t) &\equiv \frac{1}{2} \quad \text{"ignorant",}\\ \lim_{\sigma_f\to\infty}\mathrm{sig}(t) &\equiv \mathrm{step}(t) := \begin{cases} 0, t<0; & \frac{1}{2}, t=0; & 1, 0<t \end{cases} \quad \text{"hard cut".}\end{aligned}$$

In the case of very small latent scales ($\sigma_f\to 0$), the likelihood is flat causing the posterior to equal the prior. The marginal likelihood is again $Z = 2^{-n}$. (See Figure 3, regimes 7, 8 and 9.)

In the case of large latent scales ($\sigma_f \gg 1$), the likelihood approaches the step function. (See Figure 3, regimes 1, 2 and 3.) A further increase of the latent scale does not change the model anymore. The model is effectively the same for all σ_f above a threshold.

1. Furthermore, we require $g(0) = 1$ and $\lim_{t\to\infty} g(t) = 0$.

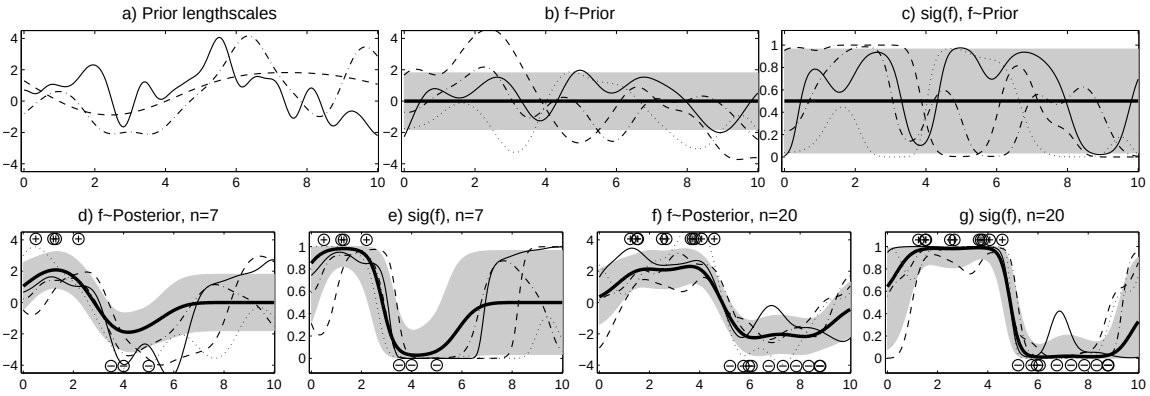


Figure 2: Pictorial illustration of binary Gaussian process classification in 1d: Plot a) shows 3 sample functions drawn from GPs with different lengthscales ℓ. Then, three pairs of plots show distributions over functions $f : \mathbb{R} \to \mathbb{R}$ and $\mathrm{sig}(f) : \mathbb{R} \to [0,1]$ occurring in GP classification. b+c) the prior, d+e) a posterior with $n = 7$ observations and f+g) a posterior with $n = 20$ observations along with the n observations with binary labels. The thick black line is the mean, the gray background is the $\pm$ standard deviation and the thin lines are sample functions. With more and more data points observed, the uncertainty is gradually shrunk. At the decision boundary the uncertainty is smallest.

2.2 Gaussian Approximations

Unfortunately, the posterior over the latent values (Equation 2) is not Gaussian due to the non-Gaussian likelihood (Equation 1). Therefore, the latent distribution (Equation 3), the predictive distribution (Equation 4) and the marginal likelihood Z cannot be written as analytical expressions. To obtain exact answers, one can resort to sampling algorithms (MCMC). However, if sig is concave in the logarithmic domain, the posterior can be shown to be unimodal motivating Gaussian approximations to the posterior. Five different Gaussian approximations corresponding to methods explained later onwards in the paper are depicted in Figure 4.

A quadratic approximation to the log likelihood $\phi(f_i) := \ln \mathbb{P}(y_i|f_i)$ at $\tilde{f}_i$

$$\phi(f_i) \;\approx\; \phi(\tilde{f}_i) + \phi'(\tilde{f}_i)(f_i - \tilde{f}_i) + \frac{1}{2}\phi''(\tilde{f}_i)(f_i - \tilde{f}_i)^2 \;=\; -\frac{1}{2} w_i f_i^2 + b_i f_i + \mathrm{const}_{f_i}$$

motivates the following approximate posterior $\mathbb{Q}(\mathbf{f}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta})$

$$\begin{aligned}
\ln \mathbb{P}(\mathbf{f}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) &\overset{(2)}{=} -\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} + \sum_{i=1}^{n} \ln \mathbb{P}(y_i|f_i) + \mathrm{const}_{\mathbf{f}} \\
&\overset{\text{quad. approx.}}{\approx} -\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\mathbf{f}^\top \mathbf{W}\mathbf{f} + \mathbf{b}^\top \mathbf{f} + \mathrm{const}_{\mathbf{f}} \\
&\overset{\mathbf{m}:=(\mathbf{K}^{-1}+\mathbf{W})^{-1}\mathbf{b}}{=} -\frac{1}{2}(\mathbf{f}-\mathbf{m})^\top \left(\mathbf{K}^{-1} + \mathbf{W}\right)(\mathbf{f}-\mathbf{m}) + \mathrm{const}_{\mathbf{f}} \\
&= \ln \mathcal{N}(\mathbf{f}|\mathbf{m}, \mathbf{V}) \;=:\; \ln \mathbb{Q}(\mathbf{f}|\mathbf{y}, \mathbf{X}, \boldsymbol{\theta}),
\end{aligned} \tag{5}$$

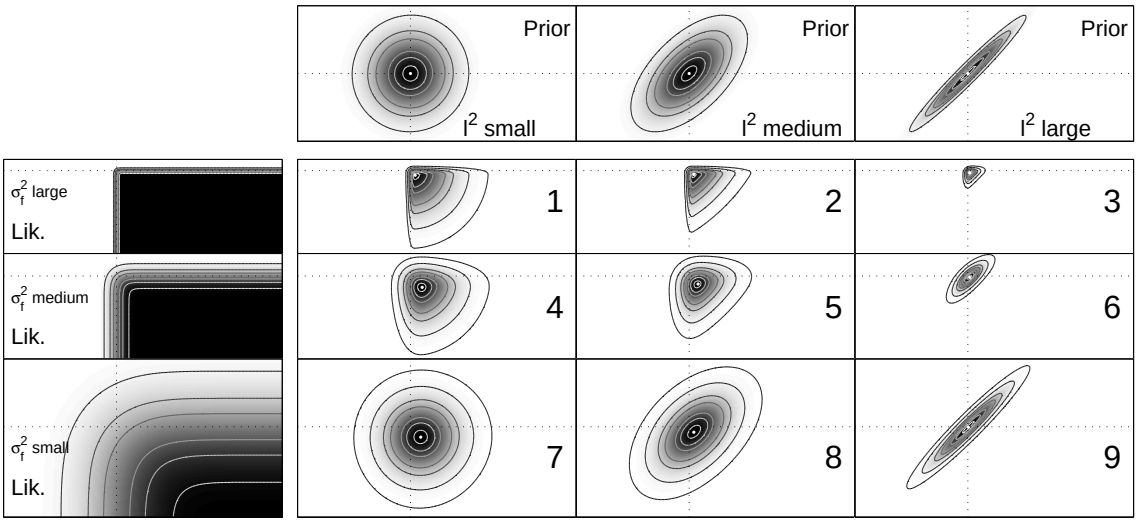


Figure 3: Gaussian Process Classification: Prior, Likelihood and exact Posterior: Nine numbered quadrants show posterior obtained by multiplication of different priors and likelihoods. The leftmost column illustrates the likelihood function for three different steepness parameters σ_f and the upper row depicts the prior for three different length scales ℓ. Here, we use σ_f as a parameter of the likelihood. Alternatively, rows correspond to "degree of Gaussianity" and columns stand for "degree of isotropy". The axes show the latent function values $f_1 = f(\mathbf{x}_1)$ and $f_2 = f(\mathbf{x}_2)$. A simple toy example employing the cumulative Gaussian likelihood and a squared exponential covariance $k(\mathbf{x},\mathbf{x}') = \sigma_f^2 \exp(-\|\mathbf{x}-\mathbf{x}'\|^2/2\ell^2)$ with length scales $\ln \ell = \{0,1,2.5\}$ and latent function scales $\ln \sigma_f = \{-1.5,0,1.5\}$ is used. Two data points $\mathbf{x}_1 = \sqrt{2}$, $\mathbf{x}_2 = -\sqrt{2}$ with corresponding labels $y_1 = 1$, $y_2 = -1$ form the data set.

where $\mathbf{V}^{-1} = \mathbf{K}^{-1} + \mathbf{W}$ and $\mathbf{W}$ denotes the precision of the effective likelihood (see Equation 7). It turns out that the methods discussed in the following sections correspond to particular choices of $\mathbf{m}$ and $\mathbf{V}$.

Let us assume, we have found such a Gaussian approximation to the posterior with mean $\mathbf{m}$ and (co)variance $\mathbf{V}$. Consequently, the latent distribution for a test point becomes a tractable one-dimensional Gaussian $\mathbb{P}(f_*|\mathbf{x}_*,\mathbf{y},\mathbf{X},\boldsymbol{\theta}) = \mathcal{N}(f_*|\mu_*,\sigma_*^2)$ with the following moments (Rasmussen and Williams, 2006, p. 44 and 56):

$$\begin{array}{rclcrcl} \mu_* & = & \mathbf{k}_*^\top \mathbf{K}^{-1}\mathbf{m} = \mathbf{k}_*^\top \boldsymbol{\alpha}, & & \boldsymbol{\alpha} & = & \mathbf{K}^{-1}\mathbf{m}, \\ \sigma_*^2 & = & k_{**} - \mathbf{k}_*^\top \left(\mathbf{K}^{-1} - \mathbf{K}^{-1}\mathbf{V}\mathbf{K}^{-1}\right)\mathbf{k}_* & = & k_{**} - \mathbf{k}_*^\top \left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\mathbf{k}_*. & & \end{array} \tag{6}$$

Since Gaussians are closed under multiplication, one can—given the Gaussian prior $\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})$ and the Gaussian approximation to the posterior $\mathbb{Q}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$—deduce the Gaussian factor $\mathbb{Q}(\mathbf{y}|\mathbf{f})$ such that $\mathbb{Q}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) \propto \mathbb{Q}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})$. Consequently, this Gaussian factor can be thought of as an *effective likelihood*. Five different effective likelihoods, corresponding to methods discussed sub-

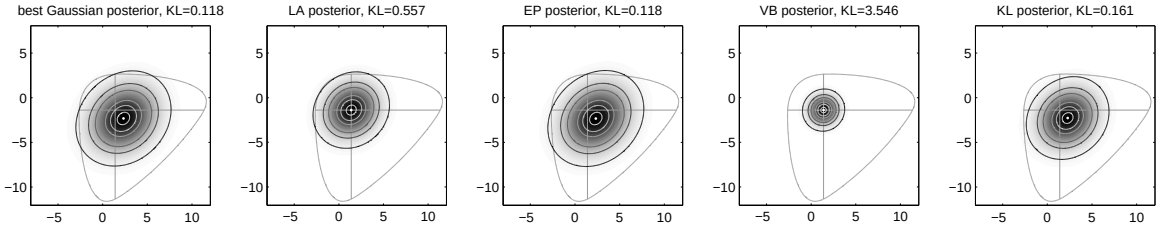


Figure 4: Five Gaussian Approximations to the Posterior (exact Posterior and mode in gray): Different Gaussian approximations to the exact posterior using the regime 2 setting of Figure 3 are shown. The exact posterior is represented in gray by a cross at the mode and a single equiprobability contour line. From left to right: The best Gaussian approximation (intractable) matches the moments of the true posterior, the Laplace approximation does a Taylor expansion around the mode, the EP approximation iteratively matches marginal moments, the variational method maximizes a lower bound on the marginal likelihood and the KL method minimizes the Kullback-Leibler to the exact posterior. The axes show the latent function values $f_1 = f(\mathbf{x}_1)$ and $f_2 = f(\mathbf{x}_2)$.

sequently in the paper, are depicted in Figure 5. By "dividing" the approximate Gaussian posterior (see Appendix B.2) by the true Gaussian prior we find the contribution of the effective likelihood $\mathbb{Q}(\mathbf{y}|\mathbf{f})$:

$$\mathbb{Q}(\mathbf{y}|\mathbf{f}) \propto \frac{\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})}{\mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})} \propto \mathcal{N}\left(\mathbf{f}|\,(\mathbf{K}\mathbf{W})^{-1}\mathbf{m}+\mathbf{m},\mathbf{W}^{-1}\right). \tag{7}$$

We see (also from Equation 5) that $\mathbf{W}$ models the precision of the effective likelihood. In general, $\mathbf{W}$ is a full matrix containing n^2 parameters.[2] However, all algorithms maintaining a Gaussian posterior approximation work with a diagonal $\mathbf{W}$ to enforce the effective likelihood to factorize over examples (as the true likelihood does, see Figure 1) in order to reduce the number of parameters. We are not aware of work quantifying the error made by this assumption.

2.3 Log Marginal Likelihood

Prior knowledge over the latent function f is encoded in the choice of a covariance function k containing hyperparameters $\boldsymbol{\theta}$. In principle, one can do inference jointly over f and $\boldsymbol{\theta}$ e.g., by sampling techniques. Another approach to model selection is maximum likelihood type II also known as the evidence framework (MacKay, 1992), where the hyperparameters $\boldsymbol{\theta}$ are chosen to maximize the marginal likelihood or evidence $\mathbb{P}(\mathbf{y}|\mathbf{X},\boldsymbol{\theta})$. In other words, one maximizes the agreement between observed data and the model. Therefore, one has a strong motivation to estimate the marginal likelihood.

Geometrically, the marginal likelihood measures the volume of the prior times the likelihood. High volume implies a strong consensus between our initial belief and our observations. In GP classification, each data point $\mathbf{x}_i$ gives rise to a dimension f_i in latent space. The likelihood implements a mechanism, for smoothly restricting the posterior along the axis of f_i to the side corresponding

2. Numerical moment matching with $\mathbf{K} = \begin{bmatrix} 7 & 6 \\ 6 & 7 \end{bmatrix}$, $y_1 = y_2 = 1$ and $\mathrm{sig}_{\mathrm{probit}}$ leads to $\mathbf{W} = \begin{bmatrix} 0.142 & -0.017 \\ -0.017 & 0.142 \end{bmatrix}$.

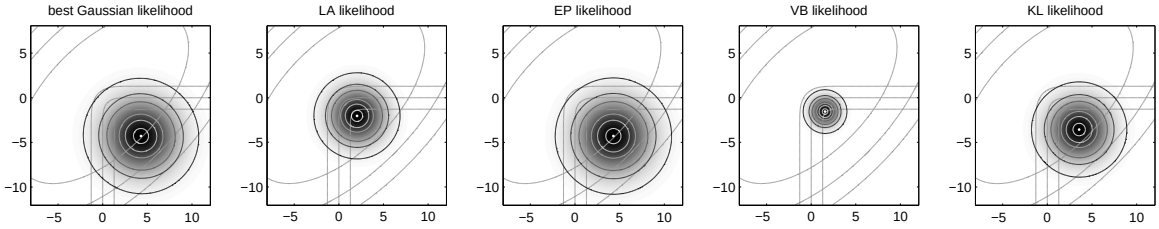


Figure 5: Five Effective Likelihoods (exact Prior/Likelihood in gray): A Gaussian approximation to the posterior induces a Gaussian effective likelihood (Equation 7). Different effective likelihoods are shown; order and setting are the same as described in Figure 4. The axes show the latent function values $f_1 = f(\mathbf{x}_1)$ and $f_2 = f(\mathbf{x}_2)$. The effective likelihood replaces the non-Gaussian likelihood (indicated by three gray lines). A good replacement behaves like the exact likelihood in regions of high prior density (indicated by gray ellipses). EP and KL yield a good coverage of that region. However LA and VB yield too concentrated replacements.

to the sign of y_i . Thus, the latent space $\mathbb{R}^n$ is softly cut down to the orthant given by the values in $\mathbf{y}$. The log marginal likelihood measures, what fraction of the prior lies in that orthant. Finally, the value $Z = 2^{-n}$ corresponds to the case, where half of the prior lies on either side along each axis in latent space. Consequently, successful inference is characterized by $Z > 2^{-n}$.

Some posterior approximations (Sections 3 and 4) provide an approximation to the marginal likelihood, other methods provide a lower bound (Sections 5 and 6). Any Gaussian approximation $\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})$ to the posterior $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$ gives rise to a lower bound Z_B to the marginal likelihood Z by application of Jensen's inequality. This bound has been used in the context of sparse approximations (Seeger, 2003).

$$\begin{aligned}\ln Z = \ln \mathbb{P}(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) \quad &= \quad \ln \int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})\,\mathrm{d}\mathbf{f} = \ln \int \mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})\,\frac{\mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})}{\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})}\mathrm{d}\mathbf{f} \\ &\overset{\text{Jensen}}{\geq} \quad \int \mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) \ln \frac{\mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})}{\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})}\mathrm{d}\mathbf{f} =: \ln Z_B. \end{aligned} \tag{8}$$

Some algebra (Appendix B.3) leads to the following expression for $\ln Z_B$:

$$\underbrace{\sum_{i=1}^{n} \int \mathcal{N}(f|,0,1) \ln \mathrm{sig}\left(y_i\left\{\sqrt{V_{ii}}f + m_i\right\}\right)\mathrm{d}f}_{\text{1) data fit}} + \frac{1}{2}[n - \underbrace{\mathbf{m}^\top \mathbf{K}^{-1}\mathbf{m}}_{\text{2) data fit}} + \underbrace{\ln\left|\mathbf{V}\mathbf{K}^{-1}\right| - \mathrm{tr}\left(\mathbf{V}\mathbf{K}^{-1}\right)}_{\text{3) regularizer}}]. \tag{9}$$

Model selection means maximization of $\ln Z_B$. Term 1) is a sum of one-dimensional Gaussian integrals of sigmoid functions in the logarithmic domain with adjustable offset and steepness. The integrals can be numerically computed in an efficient way using Gauss-Hermite quadrature (Press et al., 1993, §4.5). As the sigmoid in the log domain takes only negative values, the first term will be negative. That means, maximization of the first term is done by shifting the log-sigmoid such that the high-density region of the Gaussian is multiplied by small values. Term 2) is the equivalent

of the data-fit term in GP regression (Rasmussen and Williams, 2006, Ch. 5.4.1). Thus, the first and the second term encourage fitting the data by favouring small variances V_{ii} and large means m_i having the same sign as y_i. The third term can be rewritten as $-\ln|\mathbf{I}+\mathbf{KW}| - \mathrm{tr}\left((\mathbf{I}+\mathbf{KW})^{-1}\right)$ and yields $-\sum_{i=1}^{n}\ln(1+\lambda_i)+\frac{1}{1+\lambda_i}$ with $\lambda_i \geq 0$ being the eigenvalues of $\mathbf{KW}$. Thus, term 3) keeps the eigenvalues of $\mathbf{KW}$ small, thereby favouring a smaller class of functions—this can be seen as an instance of Occam's razor.

Furthermore, the bound

$$\ln Z_B \;=\; \int \mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) \ln \frac{\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\,\mathbb{P}(\mathbf{y}|\mathbf{X})}{\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})} \mathrm{d}\mathbf{f} \;=\; \ln Z - \mathrm{KL}\left(\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) \parallel \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\right) \qquad (10)$$

can be decomposed into the exact marginal likelihood minus the Kullback-Leibler (KL) divergence between the exact posterior and the approximate posterior. Thus by maximizing the lower bound $\ln Z_B$ on $\ln Z$, we effectively minimize the KL-divergence between $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$ and $\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})$. The bound is tight if and only if $\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) = \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$.

3. Laplace Approximation (LA)

A second order Taylor expansion around the posterior mode $\mathbf{m}$ leads to a natural way of constructing a Gaussian approximation to the log-posterior $\boldsymbol{\Psi}(\mathbf{f}) = \ln\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$ (Williams and Barber, 1998; Rasmussen and Williams, 2006, Ch. 3). The mode $\mathbf{m}$ is taken as the mean of the approximate Gaussian. Linear terms of $\boldsymbol{\Psi}$ vanish because the gradient at the mode is zero. The quadratic term of $\boldsymbol{\Psi}$ is given by the negative Hessian $\mathbf{W}$, which - due to the likelihood's factorial structure - turns out to be diagonal. The mode $\mathbf{m}$ is found by Newton's method.

3.1 Posterior

$$\begin{aligned}
\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) &\approx \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) = \mathcal{N}\left(\mathbf{f}|\mathbf{m},\left(\mathbf{K}^{-1}+\mathbf{W}\right)^{-1}\right),\\
\mathbf{m} &= \operatorname*{argmax}_{\mathbf{f}\in\mathbb{R}^n} \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta}),\\
\mathbf{W} &= -\left.\frac{\partial^2 \ln\mathbb{P}(\mathbf{y}|\mathbf{f})}{\partial\mathbf{f}\partial\mathbf{f}^\top}\right|_{\mathbf{f}=\mathbf{m}} = -\left[\left.\frac{\partial^2 \ln\mathbb{P}(y_i|f_i)}{\partial f_i^2}\right|_{f_i=m_i}\right]_{ii}.
\end{aligned}$$

3.2 Log Marginal Likelihood

The unnormalized posterior $\mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})$ has its maximum $h = \exp(\boldsymbol{\Psi}(\mathbf{m}))$ at its mode $\mathbf{m}$, where the gradient vanishes. A Taylor expansion of $\boldsymbol{\Psi}$ is then given by $\boldsymbol{\Psi}(\mathbf{f}) \approx h - \frac{1}{2}(\mathbf{f}-\mathbf{m})^\top(\mathbf{K}^{-1}+\mathbf{W})(\mathbf{f}-\mathbf{m})$. Consequently, the log marginal likelihood can be approximated by plugging in the approximation of $\boldsymbol{\Psi}(\mathbf{f})$.

$$\begin{aligned}
\ln Z = \ln\mathbb{P}(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) &= \ln\int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})\,\mathrm{d}\mathbf{f} = \ln\int \exp(\boldsymbol{\Psi}(\mathbf{f}))\,\mathrm{d}\mathbf{f}\\
&\approx \ln h + \ln\int \exp\left(-\frac{1}{2}(\mathbf{f}-\mathbf{m})^\top\left(\mathbf{K}^{-1}+\mathbf{W}\right)(\mathbf{f}-\mathbf{m})\right)\mathrm{d}\mathbf{f}\\
&= \ln\mathbb{P}(\mathbf{y}|\mathbf{m}) - \frac{1}{2}\mathbf{m}^\top\mathbf{K}^{-1}\mathbf{m} + \frac{1}{2}\ln|\mathbf{I}+\mathbf{KW}|.
\end{aligned}$$

4. Expectation Propagation (EP)

EP (Minka, 2001b) is an iterative method to find approximations based on approximate marginal moments, which can be applied to Gaussian processes. See (Rasmussen and Williams, 2006, Ch. 3) for details. The individual likelihood terms are replaced by site functions $t_i(f_i)$ being unnormalized Gaussians

$$\mathbb{P}(y_i|f_i) \approx t_i\left(f_i,\mu_i,\sigma_i^2,Z_i\right) := Z_i\mathcal{N}\left(f_i|\mu_i,\sigma_i^2\right)$$

such that the approximate marginal moments of $\mathbb{Q}(f_i) := \int \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})\prod_{j=1}^n Z_j\mathcal{N}\left(f_j|\mu_j,\sigma_j^2\right)\mathrm{d}\mathbf{f}_{\neg i}$ agree with the marginals of $\int \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})\mathbb{P}(y_i|f_i)\prod_{j\neq i} Z_j\mathcal{N}\left(f_j|\mu_j,\sigma_j^2\right)\mathrm{d}\mathbf{f}_{\neg i}$ of the approximation based on the exact likelihood term $\mathbb{P}(y_j|f_j)$. That means, there are $3n$ quantities μ_i, σ_i^2 and Z_i to be iteratively optimized. Convergence of EP is not generally guaranteed, but there always exists a fixed-point for the EP updates in GP classification (Minka, 2001a). If the EP iterations converge, the solution obtained is a saddle point of a special energy function (Minka, 2001a). However, an EP update does not necessarily imply a decrease in energy. For our case of log-concave likelihood functions, we always observed convergence, but we are not aware of a formal proof.

4.1 Posterior

Based on these local approximations, the approximate posterior can be written as:

$$\begin{aligned}
\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) &\approx \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) = \mathcal{N}\left(\mathbf{f}|\mathbf{m},\left(\mathbf{K}^{-1}+\mathbf{W}\right)^{-1}\right),\\
\mathbf{W} &= \left[\sigma_i^{-2}\right]_{ii},\\
\mathbf{m} &= \mathbf{V}\mathbf{W}\boldsymbol{\mu} = \left[\mathbf{I}-\mathbf{K}\left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\right]\mathbf{K}\mathbf{W}\boldsymbol{\mu},\ \boldsymbol{\mu}=(\mu_1,\ldots,\mu_n)^\top.
\end{aligned}$$

4.2 Log Marginal Likelihood

>From the likelihood approximations, one can directly obtain an expression for the approximate log marginal likelihood

$$\begin{aligned}
\ln Z = \ln\mathbb{P}(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) &= \ln\int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})\,\mathrm{d}\mathbf{f}\\
&\approx \ln\int\prod_{i=1}^n t\left(f_i,\mu_i,\sigma_i^2,Z_i\right)\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})\,\mathrm{d}\mathbf{f}\\
&= \sum_{i=1}^n \ln Z_i - \frac{1}{2}\boldsymbol{\mu}^\top\left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\boldsymbol{\mu} - \frac{1}{2}\ln\left|\mathbf{K}+\mathbf{W}^{-1}\right| - \frac{n}{2}\ln 2\pi\\
&= \sum_{i=1}^n \ln\frac{Z_i}{\sqrt{2\pi}} - \frac{1}{2}\mathbf{m}^\top\left(\mathbf{K}^{-1}+\mathbf{K}^{-1}\mathbf{W}^{-1}\mathbf{K}^{-1}\right)\mathbf{m} - \frac{1}{2}\ln\left|\mathbf{K}+\mathbf{W}^{-1}\right| =: \ln Z_{EP}.
\end{aligned}$$

The lower bound provided by Jensen's inequality Z_B (Equation 9) is known to be below the approximation Z_{EP} obtained by EP (Opper and Winther, 2005, page 2183). From $Z_{EP} \geq Z_B$ and $Z \geq Z_B$ it is not clear, which value one should use. In principle, Z_{EP} could be a bad approximation. However, our experimental findings and extensive Monte Carlo simulations suggest that Z_{EP} is very accurate.

4.3 Thouless, Anderson & Palmer method (TAP)

Based on ideas rooted in Statistical Physics, one can approach the problem from a slightly different angle (Opper and Winther, 2000). Individual Gaussian approximations $\mathcal{N}(f_i|\mu_{\neg i},\sigma^2_{\neg i})$ are only made to predictive distributions $\mathbb{P}\left(f_i|\mathbf{x}_i,\mathbf{y}_{\setminus i},\mathbf{X}_{\setminus i},\boldsymbol{\theta}\right)$ for data points $\mathbf{x}_i$ that have been previously removed from the training set. Based on $\mu_{\neg i}$ and $\sigma^2_{\neg i}$ one can derive explicit expressions for $(\boldsymbol{\alpha},\mathbf{W}^{\frac{1}{2}})$, our parameters of interest.

$$\begin{aligned}\alpha_i &\approx \frac{\int \frac{\partial}{\partial f_i}\mathbb{P}(y_i|f_i)\,\mathcal{N}(f_i|\mu_{\neg i},\sigma^2_{\neg i})\mathrm{d}f_i}{\int \mathbb{P}(y_i|f_i)\,\mathcal{N}(f_i|\mu_{\neg i},\sigma^2_{\neg i})\mathrm{d}f_i},\\ \left[\mathbf{W}^{-1}\right]_{ii} &\approx \sigma^2_{\neg i}\left(\frac{1}{\alpha_i\left[\mathbf{K}\boldsymbol{\alpha}\right]_i}-1\right).\end{aligned} \tag{11}$$

In turn, the $2n$ parameters $(\mu_{\neg i},\sigma^2_{\neg i})$ can be expressed as a function of $\boldsymbol{\alpha}$, $\mathbf{K}$ and $\mathbf{W}^{\frac{1}{2}}$.

$$\begin{aligned}\sigma^2_{\neg i} &= 1/\left[\left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\right]_{ii}-\left[\mathbf{W}^{-1}\right]_{ii},\\ \mu_{\neg i} &= \left[\mathbf{K}\boldsymbol{\alpha}\right]_i-\sigma^2_{\neg i}\alpha_i.\end{aligned} \tag{12}$$

As a result, a system (Equations 11/12) of nonlinear equations in $\mu_{\neg i}$ and $\sigma^2_{\neg i}$ has to be solved by iteration. Each step involves a matrix inversion of cubic complexity. A faster "naïve" variant updating only n parameters has also been proposed (Opper and Winther, 2000) but it does not lead to the same fixed point. As in the FV algorithm (Section 7), a formal complex transformation leads to a simplified version by fixing $\sigma^2_{\neg i}=\mathbf{K}_{ii}$, called (TAPnaive) in the sequel.

Finally, for prediction, the predictive posterior $\mathbb{P}(f_*|\mathbf{x}_*,\mathbf{y},\mathbf{X},\boldsymbol{\theta})$ is approximated by a Gaussian $\mathcal{N}(f_*|\mu_*,\sigma^2_*)$ at a test point $\mathbf{x}_*$ based on the parameters $(\boldsymbol{\alpha},\mathbf{W}^{\frac{1}{2}})$ and according to equation (6).

A fixed-point of the TAP mean-field equations is also a fixed-point of the EP algorithm (Minka, 2001a). This theoretical result was confirmed in our numerical simulations. However, the EP algorithm is more practical and typically much faster. For this reason, we are not going to treat the TAP method as an independent algorithm in this paper.

5. KL-Divergence Minimization (KL)

In principle, we simply want to minimize a dissimilarity measure between the approximate posterior $\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})=\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})$ and the exact posterior $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$. One quantity to minimize is the KL-divergence

$$\mathrm{KL}\left(\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\parallel\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})\right) = \int\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\ln\frac{\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})}{\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})}\mathrm{d}\mathbf{f}.$$

Unfortunately, this expression is intractable. If instead, we measure the reverse KL-divergence, we regain tractability

$$\mathrm{KL}\left(\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta})\parallel\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\right) = \int\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\ln\frac{\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})}{\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})}\mathrm{d}\mathbf{f}=:\mathrm{KL}(\mathbf{m},\mathbf{V}).$$

A similar approach has been followed for regression with Laplace or Cauchy noise (Opper and Archambeau, 2008). Finally, we minimize the following objective (see Appendix B.3) with respect to the variables $\mathbf{m}$ and $\mathbf{V}$. Constant terms have been dropped from the expression:

$$\mathrm{KL}(\mathbf{m},\mathbf{V}) \stackrel{c}{=} -\int \mathcal{N}(f)\left[\sum_{i=1}^{n} \ln \mathrm{sig}\left(\sqrt{v_{ii}}y_i f + m_i y_i\right)\right] \mathrm{d}f - \frac{1}{2}\ln|\mathbf{V}| + \frac{1}{2}\mathbf{m}^\top \mathbf{K}^{-1}\mathbf{m} + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{V}\right).$$

We refer to the first term of $\mathrm{KL}(\mathbf{m},\mathbf{V})$ as $a(\mathbf{m},\mathbf{V})$ to keep the expressions short. We calculate first derivatives and equate them with zero to obtain necessary conditions that have to be fulfilled at a local optimum $(\mathbf{m}^*,\mathbf{V}^*)$

$$\begin{aligned}
\frac{\partial \mathrm{KL}}{\partial \mathbf{m}} &= \frac{\partial a}{\partial \mathbf{m}} - \mathbf{K}^{-1}\mathbf{m} = \mathbf{0} \;\Rightarrow\; \mathbf{K}^{-1}\mathbf{m} = \frac{\partial a}{\partial \mathbf{m}} = \boldsymbol{\alpha}, \\
\frac{\partial \mathrm{KL}}{\partial \mathbf{V}} &= \frac{\partial a}{\partial \mathbf{V}} + \frac{1}{2}\mathbf{V}^{-1} - \frac{1}{2}\mathbf{K}^{-1} = \mathbf{0} \;\Rightarrow\; \mathbf{V} = \left(\mathbf{K}^{-1} - 2\frac{\partial a}{\partial \mathbf{V}}\right)^{-1} = \left(\mathbf{K}^{-1} - 2\boldsymbol{\Lambda}\right)^{-1}
\end{aligned}$$

which defines $\boldsymbol{\Lambda}$. If the approximate posterior is parametrized by $(\mathbf{m},\mathbf{V})$, there are in principle in the order of n^2 parameters. But if the necessary conditions for a local minimum are fulfilled (i.e., the derivatives $\partial\mathrm{KL}/\partial\mathbf{m}$ and $\partial\mathrm{KL}/\partial\mathbf{V}$ vanish), the problem can be re-parametrized in terms of $(\boldsymbol{\alpha},\boldsymbol{\Lambda})$. Since $\boldsymbol{\Lambda} = \partial a/\partial\mathbf{V}$ is a diagonal matrix (see Equation 17), the optimum is characterized $2n$ free parameters. This fact was already pointed out by Manfred Opper (personal communication) and Matthias Seeger (Seeger, 1999, Ch. 5.21, Eq. 5.3). Thus, a minimization scheme based on Newton iterations on the joint vector $\boldsymbol{\xi} := [\boldsymbol{\alpha}^\top, \boldsymbol{\Lambda}_{ii}]^\top$ takes $O(8\cdot n^3)$ operations. Details about the derivatives $\partial\mathrm{KL}/\partial\boldsymbol{\xi}$ and $\partial^2\mathrm{KL}/\partial\boldsymbol{\xi}\partial\boldsymbol{\xi}^\top$ are provided in Appendix A.2.

5.1 Posterior

Based on these local approximations, the approximate posterior can be written as:

$$\begin{aligned}
\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) &\approx \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) = \mathcal{N}\left(\mathbf{f}|\mathbf{m},\left(\mathbf{K}^{-1}+\mathbf{W}\right)^{-1}\right), \\
\mathbf{W} &= -2\boldsymbol{\Lambda}, \\
\mathbf{m} &= \mathbf{K}\boldsymbol{\alpha}.
\end{aligned}$$

5.2 Log Marginal Likelihood

Since the method inherently maximizes a lower bound on the marginal likelihood, this bound (Equation 9) is used as approximation to the marginal likelihood.

6. Variational Bounds (VB)

The following variational bounding method (Gibbs and MacKay, 2000) is a special case of the KL method. Instead of optimizing a bound on the joint (Eq. 8), they impose the bounding condition on each likelihood term individually. Here, we treat parametrization based on quadratic lower bounds on the individual likelihoods in the logarithmic domain. We first derive all calculations based on

general likelihoods. Individual likelihood bounds

$$\begin{aligned}\mathbb{P}(y_i|f_i) &\geq \exp\left(a_i f_i^2 + b_i y_i f_i + c_i\right), \forall f_i \in \mathbb{R} \forall i \\ \Rightarrow \mathbb{P}(\mathbf{y}|\mathbf{f}) &\geq \exp\left(\mathbf{f}^\top \mathbf{A}\mathbf{f} + (\mathbf{b}\odot\mathbf{y})^\top \mathbf{f} + \mathbf{c}^\top \mathbb{1}\right) =: \mathbb{Q}(\mathbf{y}|\mathbf{f},\mathbf{A},\mathbf{b},\mathbf{c}), \forall \mathbf{f} \in \mathbb{R}\end{aligned}$$

are defined in terms of coefficients a_i, b_i and c_i, where $\odot$ denotes the element-wise product of two vectors. This lower bound on the likelihood induces a lower bound on the marginal likelihood.

$$Z = \int \mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathrm{d}\mathbf{f} \quad \geq \quad \int \mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathbb{Q}(\mathbf{y}|\mathbf{f},\mathbf{A},\mathbf{b},\mathbf{c})\,\mathrm{d}\mathbf{f} = Z_B.$$

Carrying out the Gaussian integral

$$Z_B \quad = \quad \int \mathcal{N}(\mathbf{f}|0,\mathbf{K}) \exp\left(\mathbf{f}^\top \mathbf{A}\mathbf{f} + (\mathbf{b}\odot\mathbf{y})^\top \mathbf{f} + \mathbf{c}^\top \mathbb{1}\right) \mathrm{d}\mathbf{f}$$

leads to (see Appendix B.4)

$$\ln Z_B \quad = \quad \mathbf{c}^\top \mathbb{1} + \frac{1}{2}(\mathbf{b}\odot\mathbf{y})^\top \left(\mathbf{K}^{-1} - 2\mathbf{A}\right)^{-1} (\mathbf{b}\odot\mathbf{y}) - \frac{1}{2}\ln|\mathbf{I} - 2\mathbf{A}\mathbf{K}| \tag{13}$$

which can now be maximized with respect to the coefficients a_i, b_i and c_i. In order to get an efficient algorithm, one has to calculate the first and second derivatives $\partial \ln Z_B/\partial \boldsymbol{\varsigma}$, $\partial^2 \ln Z_B/\partial \boldsymbol{\varsigma}\partial \boldsymbol{\varsigma}^\top$ (as done in Appendix A.1). Hyperparameters can be optimized using the gradient $\partial \ln Z_B/\partial \boldsymbol{\theta}$.

6.1 Logit Bound

Optimizing the logistic likelihood function (Gibbs and MacKay, 2000), we obtain the necessary conditions

$$\begin{aligned}\mathbf{A}_{\boldsymbol{\varsigma}} &:= -\boldsymbol{\Lambda}_{\boldsymbol{\varsigma}}, \\ \mathbf{b}_{\boldsymbol{\varsigma}} &:= \frac{1}{2}\mathbb{1}, \\ \mathbf{c}_{\boldsymbol{\varsigma},i} &:= \varsigma_i^2 \lambda(\varsigma_i) - \frac{1}{2}\varsigma_i + \ln \mathrm{sig}_{\mathrm{logit}}(\varsigma_i)\end{aligned}$$

where we define $\lambda(\varsigma_i) = \left(2\mathrm{sig}_{\mathrm{logit}}(\varsigma_i) - 1\right)/(4\varsigma_i)$ and $\boldsymbol{\Lambda}_{\boldsymbol{\varsigma}} = [\lambda(\varsigma_i)]_{ii}$. This shows, that we only have to optimize with respect to n parameters $\boldsymbol{\varsigma}$. We apply Newton's method for this purpose. The bound is symmetric and tight at $\mathbf{f} = \pm\boldsymbol{\varsigma}$.

6.2 Probit Bound

For reasons of completeness, we derive similar expressions (Appendix B.5) for the cumulative Gaussian likelihood $\mathrm{sig}_{\mathrm{probit}}(f_i)$ with necessary conditions

$$\begin{aligned}\mathbf{a}_{\boldsymbol{\varsigma}} &:= -\frac{1}{2}\mathbb{1}, \\ \mathbf{b}_{\boldsymbol{\varsigma},i} &:= \varsigma_i + \frac{\mathcal{N}(\varsigma_i)}{\mathrm{sig}_{\mathrm{probit}}(\varsigma_i)}, \\ \mathbf{c}_{\boldsymbol{\varsigma},i} &:= \left(\frac{\varsigma_i}{2} - b_i\right)\varsigma_i + \ln\left(\mathrm{sig}_{\mathrm{probit}}(\varsigma_i)\right)\end{aligned} \tag{14}$$

which again depend only on a single vector of parameters we optimize using Newton's method. The bound is tight for $\mathbf{f} = \boldsymbol{\varsigma}$.

6.3 Posterior

Based on these local approximations, the approximate posterior can be written as

$$\begin{aligned}\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) &\approx \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) = \mathcal{N}\left(\mathbf{f}|\mathbf{m},\left(\mathbf{K}^{-1}+\mathbf{W}\right)^{-1}\right),\\ \mathbf{W} &= -2\mathbf{A}_{\boldsymbol{\varsigma}},\\ \mathbf{m} &= \mathbf{V}(\mathbf{y}\odot\mathbf{b}_{\boldsymbol{\varsigma}}) = \left(\mathbf{K}^{-1}-2\mathbf{A}_{\boldsymbol{\varsigma}}\right)^{-1}(\mathbf{y}\odot\mathbf{b}_{\boldsymbol{\varsigma}}),\end{aligned}$$

where we have expressed the posterior parameters directly as a function of the coefficients. Finally, we deal with an approximate posterior $\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) = \mathcal{N}(\mathbf{f}|\mathbf{m}_{\boldsymbol{\varsigma}},\mathbf{V}_{\boldsymbol{\varsigma}})$ only depending on a vector $\boldsymbol{\varsigma}$ of n variational parameters and a mapping $\boldsymbol{\varsigma}\mapsto(\mathbf{m}_{\boldsymbol{\varsigma}},\mathbf{V}_{\boldsymbol{\varsigma}})$. In the KL method, every combination of values $\mathbf{m}$ and $\mathbf{W}$ is allowed, in the VB method, $\mathbf{m}_{\boldsymbol{\varsigma}}$ and $\mathbf{V}_{\boldsymbol{\varsigma}}$ cannot be chosen independently, since the have to be compatible with the bounding requirements. Therefore, the variational posterior is more constrained than the general Gaussian posterior and thus easier to optimize.

6.4 Log Marginal Likelihood

It turns out, that the approximation to the marginal likelihood (Equation 13) is often quite poor and the more general Jensen bound approach (Equation 9) is much tighter. In practice, one would have to evaluate both of them and keep the maximum value.

7. Factorial Variational Method (FV)

Instead of approximating the posterior $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})$ by the closest Gaussian distribution, one can use the closest factorial distribution $\mathbb{Q}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) = \prod_i \mathbb{Q}(f_i)$, also called *ensemble learning* (Csató et al., 2000). Another kind of factorial approximation $\mathbb{Q}(\mathbf{f}) = \mathbb{Q}(\mathbf{f}^+)\mathbb{Q}(\mathbf{f}^-)$—a posterior factorizing over classes—is used in multi-class classification (Girolami and Rogers, 2006).

7.1 Posterior

As a result of free-form minimization of the Kullback-Leibler divergence $\mathrm{KL}(\mathbb{Q}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}) \parallel \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta}))$ by equating its functional derivative $\delta\mathrm{KL}/\delta\mathbb{Q}(f_i)$ with the zero function (Appendix B.6), one finds the best approximation to be of the following form:

$$\begin{aligned}\mathbb{Q}(f_i) &\propto \mathcal{N}\left(f_i\,|\mu_i,\sigma_i^2\right)\mathbb{P}(y_i|f_i),\\ \mu_i &= m_i - \sigma_i^2\left[\mathbf{K}^{-1}\mathbf{m}\right]_i = [\mathbf{K}\boldsymbol{\alpha}]_i - \sigma_i^2\alpha_i,\\ \sigma_i^2 &= \left[\mathbf{K}^{-1}\right]_{ii}^{-1},\\ m_i &= \int f_i\mathbb{Q}(f_i)\,\mathrm{d}f_i.\end{aligned} \tag{15}$$

In fact, the best product distribution consists of a factorial Gaussian times the original likelihood. The Gaussian has the same moments as the Leave-One-Out prediction (Sundararajan and Keerthi, 2001). Since the posterior is factorial, the effective likelihood of the factorial approximation has an odd shape. It effectively has to annihilate the correlations in the prior, and these correlations are usually what allows learning to happen in the first place. However, the best fitting factorial is still able to ensure that the latent means have the right signs. Even though all correlations are neglected,

it is still possible that the model picks up the most important structure, since the expectations are coupled. Of course, at test time, it is essential that correlations are taken into account again using Equation 6, as it would otherwise be impossible to inject any knowledge into the predictive distribution. For predictions we use the Gaussian $\mathcal{N}(\mathbf{f}|\mathbf{m},\mathrm{Dg}(\mathbf{v}))$ instead of $\mathbb{Q}(\mathbf{f})$. This is a further approximation, but it allows to stay inside the Gaussian framework.

Parameters μ_i and m_i are found by the following algorithm. Starting from $\mathbf{m}=\mathbf{0}$, iterate the following until convergence; (1) compute μ_i, (2) update m_i by taking a step in the direction towards m_i as given by Equation 15. Stepsizes are adapted.

7.2 Log Marginal Likelihood

Surprisingly, one can obtain a lower bound on the marginal likelihood (Csató et al., 2000):

$$\ln Z \quad \geq \quad \sum_{i=1}^{n} \ln \mathrm{sig}\left(\frac{y_i m_i}{\sigma_i}\right) - \frac{1}{2}\boldsymbol{\alpha}^\top \left(\mathbf{K} - \mathrm{Dg}(\left[\sigma_1^2,\ldots,\sigma_n^2\right]^\top)\right)\boldsymbol{\alpha} - \frac{1}{2}\ln|\mathbf{K}| + \sum_{i=1}^{n}\ln\sigma_i.$$

8. Label Regression Method (LR)

Classification has also been treated using label regression or least squares classification (Rifkin and Klautau, 2004). In its simplest form, this method simply ignores the discreteness of the class labels at the cost of not being able to provide proper probabilistic predictions. However, we treat LR as a heuristic way of choosing $\boldsymbol{\alpha}$ and $\mathbf{W}$, which allows us to think of it as yet another Gaussian approximation to the posterior allowing for valid predictions of class probabilities.

8.1 Posterior

After inference, according to Equation 6, the moments of the (Gaussian approximation to the) posterior GP can be written as $\mu_* = \mathbf{k}_*^\top \boldsymbol{\alpha}$ and $\sigma_*^2 = k_{**} - \mathbf{k}_*^\top \left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\mathbf{k}_*$. Fixing

$$\mathbf{W}^{-1} = \sigma_n^2\mathbf{I} \quad \text{and} \quad \boldsymbol{\alpha} = \left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\left(\mathbf{K}+\mathbf{W}^{-1}\right)\boldsymbol{\alpha} = \left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\mathbf{y},$$

we obtain GP regression from data points $\mathbf{x}_i \in \mathcal{X}$ to real labels $y_i \in \mathbb{R}$ with noise of variance σ_n^2 as a special case. In regression, the posterior moments are given by $\mu_* = \mathbf{k}_*^\top \left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{y}$ and $\sigma_*^2 = k_{**} - \mathbf{k}_*^\top \left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)^{-1}\mathbf{k}_*$ (Rasmussen and Williams, 2006). The arbitrary scale of the discrete $\mathbf{y}$ can be absorbed by the hyperparameters. There is an additional parameter σ_n, describing the width of the effective likelihood. In experiments, we selected $\sigma_n \in [0.5, 2]$ to maximise the log marginal likelihood.

8.2 Log Marginal Likelihood

There are two ways of obtaining an estimate of the log marginal likelihood. One can simply ignore the binary nature and use the regression marginal likelihood $\ln Z_{\mathrm{reg}}$ as proxy for $\ln Z$—an approach we only mention but not use in the experiments

$$\ln Z_{\mathrm{reg}} \quad = \quad -\frac{1}{2}\boldsymbol{\alpha}^\top\left(\mathbf{K}+\sigma_n^2\mathbf{I}\right)\boldsymbol{\alpha} - \frac{1}{2}\ln\left|\mathbf{K}+\sigma_n^2\mathbf{I}\right| - \frac{n}{2}\ln 2\pi.$$

Alternatively, the Jensen bound (8) yields a lower bound $\ln Z \geq \ln Z_B$—which seems more in line with the classification scenario than $\ln Z_{\mathrm{reg}}$.

9. Relations Between the Methods

All considered approximations can be separated into *local* and *global methods*. Local methods exploit properties (such as derivatives) of the posterior at a special location only. Global methods minimize the KL-divergence $\mathrm{KL}(\mathbb{Q}||\mathbb{P}) = \int \mathbb{Q}(\mathbf{f}) \ln \mathbb{Q}(\mathbf{f}) / \mathbb{P}(\mathbf{f}) \, \mathrm{d}\mathbf{f}$ between the posterior $\mathbb{P}(\mathbf{f})$ and a tractable family of distributions $\mathbb{Q}(\mathbf{f})$. Often this methodology is also referred to as a variational algorithm.

assumption	relation	conditions	approx. posterior $\mathbb{Q}(\mathbf{f})$	name
$\mathbb{Q}(\mathbf{f}) = \mathcal{N}(\mathbf{f}\|\mathbf{m},\mathbf{V})$	$\rightarrow$	$\mathbf{m} = \operatorname{argmax}_{\mathbf{f}} \mathbb{P}(\mathbf{f})$ $\mathbf{W} = -\frac{\partial^2 \ln \mathbb{P}(\mathbf{y}\|\mathbf{f})}{\partial \mathbf{f} \partial \mathbf{f}^\top}$	$\mathcal{N}(\mathbf{f}\|\mathbf{m},(\mathbf{K}^{-1}+\mathbf{W})^{-1})$	LA
$\mathbb{Q}(\mathbf{f}) = \prod_i q_i(f_i)$	$\rightarrow$	$\frac{\delta \mathrm{KL}}{\delta q_i(f_i)} \equiv \mathbf{0}$	$\prod_i \mathcal{N}(f_i\|\mu_i,\sigma_i^2)\mathbb{P}(y_i\|f_i)$	FV
	$\searrow$			
		$\left\langle f_i^d \right\rangle_{q_i(f_i)} = \left\langle f_i^d \right\rangle_{\mathbb{Q}(f_i)}$	$\mathcal{N}\left(\mathbf{f}\|\mathbf{m},(\mathbf{K}^{-1}+\mathbf{W})^{-1}\right)$	EP
	$\nearrow$			
$\mathbb{Q}(\mathbf{f}) = \mathcal{N}(\mathbf{f}\|\mathbf{m},\mathbf{V})$	$\rightarrow$	$\frac{\partial \mathrm{KL}}{\partial \mathbf{V},\mathbf{m}} = \mathbf{0}$	$\mathcal{N}\left(\mathbf{f}\|\mathbf{m},(\mathbf{K}^{-1}+\mathbf{W})^{-1}\right)$	KL
	$\searrow$			
$\mathbb{P}(y_i\|f_i) \geq \mathcal{N}(f_i\|\mu_{\varsigma i},\sigma_{\varsigma i}^2)$	$\rightarrow$	$\frac{\partial \mathrm{KL}}{\partial \boldsymbol{\varsigma}_*} = \mathbf{0}$	$\mathcal{N}\left(\mathbf{f}\|\mathbf{m}_{\boldsymbol{\varsigma}_*},(\mathbf{K}^{-1}+\mathbf{W}_{\boldsymbol{\varsigma}_*})^{-1}\right)$	VB
$\mathbb{P}(y_i\|f_i) := \mathcal{N}(f_i\|y_i,\sigma_n^2)$	$\rightarrow$	$\mathbf{m} = (\mathbf{I}+\sigma_n^2\mathbf{K}^{-1})^{-1}\mathbf{y}$	$\mathcal{N}(\mathbf{f}\|\mathbf{m},(\mathbf{K}^{-1}+\sigma_n^{-2}\mathbf{I})^{-1})$	LR

The only local method considered is the LA approximation matching curvature at the posterior mode. Common tractable distributions for global methods include factorial and Gaussian distributions. They have their direct correspondent in the FV method and the KL method. Individual likelihood bounds make the VB method a more constrained and easier-to-optimize version of the KL method. Interestingly, EP can be seen in some sense as a hybrid version of FV and KL, combining the advantages of both methods. Within the Expectation Consistence framework (Opper and Winther, 2005), EP can be thought of as an algorithm that implicitly works with two distributions—a factorial and a Gaussian—having the same marginal moments $\left\langle f_i^d \right\rangle$. By means of iterative updates, one keeps these expectations consistent and produces a posterior approximation.

In the divergence measure and message passing framework (Minka, 2005), EP is cast as a message passing algorithm template: Iterative minimization of local divergences to a tractable family of distributions yields a small global divergence. From that viewpoint, FV and KL are considered as special cases with divergence measure $\mathrm{KL}(\mathbb{Q}||\mathbb{P})$ combined with factorial and Gaussian distributions.

There is also a link between local and global methods, namely from the KL to the LA method. The necessary conditions for the LA method do hold *on average* for the KL method (Opper and Archambeau, 2008).

Finally, LR neither qualifies as local nor global—it is just a heuristic way of setting $\mathbf{m}$ and $\mathbf{W}$.

10. Markov Chain Monte Carlo (MCMC)

The only way of getting a handle on the ground truth for the moments Z, $\mathbf{m}$ and $\mathbf{V}$ is by applying sampling techniques. In the limit of long runs, one is guaranteed to get the right answer. But in practice, these methods can be very slow, compared to analytic approximations discussed previously. MCMC runs are rather supposed to provide a gold standard for the comparison of the other methods.

It turns out to be most challenging to obtain reliable marginal likelihood estimates as it is equivalent to solving the free energy problem in physics. We employ Annealed Importance Sampling (AIS) and thermodynamic integration to yield the desired marginal likelihoods. Instead of starting annealing from the prior distribution, we propose to directly start from an approximate posterior in order to speed up the sampling process.

Accurate estimates of the first and second moments can be obtained by sampling directly from the (unnormalized) posterior using Hybrid Monte Carlo methods (Neal, 1993).

10.1 Thermodynamic Integration

The goal is to calculate the marginal likelihood $Z = \int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}$. AIS (Neal, 1993, 2001) works with intermediate quantities $Z_t := \int \mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t)}\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}$. Here, $\tau : \mathbb{N} \supset [0,T] \to [0,1] \subset \mathbb{R}$ denotes an inverse temperature schedule with the properties $\tau(0) = 0$, $\tau(T) = 1$ and $\tau(t+1) \geq \tau(t)$ leading to $Z_0 = \int \mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f} = 1$ and $Z_T = Z$.

On the other hand, we have $Z = Z_T/Z_0 = \prod_{t=1}^{T} Z_t/Z_{t-1}$—an expanded fraction. Each factor Z_t/Z_{t-1} can be approximated by importance sampling with samples $\mathbf{f}_s$ from the "intermediate posterior" $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1) := \mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t-1)}\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,/Z_{t-1}$ at time t.

$$\begin{aligned}
\frac{Z_t}{Z_{t-1}} &= \frac{\int \mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t)}\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}}{Z_{t-1}} = \int \frac{\mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t)}}{\mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t-1)}} \frac{\mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t-1)}\,\mathbb{P}(\mathbf{f}|\mathbf{X})}{Z_{t-1}}\mathrm{d}\mathbf{f} \\
&= \int \mathbb{P}(\mathbf{y}|\mathbf{f})^{\Delta\tau(t)}\,\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1)\,\mathrm{d}\mathbf{f} \\
&\approx \frac{1}{S}\sum_{s=1}^{S} \mathbb{P}(\mathbf{y}|\mathbf{f}_s)^{\Delta\tau(t)}, \qquad \mathbf{f}_s \sim \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1)\,.
\end{aligned}$$

This works fine for small temperature changes $\Delta\tau(t) := \tau(t) - \tau(t-1)$. In the limit, we smoothly interpolate between $\mathbb{P}(\mathbf{y}|\mathbf{f})^0\,\mathbb{P}(\mathbf{f}|\mathbf{X})$ and $\mathbb{P}(\mathbf{y}|\mathbf{f})^1\,\mathbb{P}(\mathbf{f}|\mathbf{X})$, that is, we start by sampling from the prior and finally approach the posterior. Note that sampling is algorithmically possible even though the distribution is only known up to a constant factor.

10.2 Amelioration Using an Approximation to the Posterior

In practice, the posterior can be quite different from the prior. That means that individual fractions Z_t/Z_{t-1} may be difficult to estimate. One can make these fractions more similar by increasing the number of steps T or by "starting" from a distribution close to the posterior rather than from the prior. Let $\mathbb{Q}(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) \approx \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},T) = \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,/Z_T$ denote an approximation to the posterior. Setting $\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) = \mathbb{Q}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X})$, one can calculate the effective likelihood $\mathbb{Q}(\mathbf{y}|\mathbf{f})$ by division (see Appendix B.2).

For the integration we use $Z_t = \int \mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t)}\,\mathbb{Q}(\mathbf{y}|\mathbf{f})^{1-\tau(t)}\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}$ where $Z_0 = \int \mathbb{Q}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}$ can be computed analytically. Again, each factor $\frac{Z_t}{Z_{t-1}}$ of the expanded fraction can be approximated

by importance sampling from the modified intermediate posterior:

$$\begin{aligned}\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1) &= \mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t-1)}\mathbb{Q}(\mathbf{y}|\mathbf{f})^{1-\tau(t-1)}\mathbb{P}(\mathbf{f}|\mathbf{X})/Z_{t-1} \\ &= \left[\frac{\mathbb{P}(\mathbf{y}|\mathbf{f})}{\mathbb{Q}(\mathbf{y}|\mathbf{f})}\right]^{\tau(t-1)}\mathbb{Q}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X})/Z_{t-1},\end{aligned}$$

$$\begin{aligned}\frac{Z_t}{Z_{t-1}} &= \frac{\int \mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t)}\mathbb{Q}(\mathbf{y}|\mathbf{f})^{1-\tau(t)}\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}}{Z_{t-1}} \\ &= \int \frac{\mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t)}\mathbb{Q}(\mathbf{y}|\mathbf{f})^{1-\tau(t)}}{\mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t-1)}\mathbb{Q}(\mathbf{y}|\mathbf{f})^{1-\tau(t-1)}}\frac{\mathbb{P}(\mathbf{y}|\mathbf{f})^{\tau(t-1)}\mathbb{Q}(\mathbf{y}|\mathbf{f})^{1-\tau(t-1)}\mathbb{P}(\mathbf{f}|\mathbf{X})}{Z_{t-1}}\mathrm{d}\mathbf{f} \\ &= \int \left[\frac{\mathbb{P}(\mathbf{y}|\mathbf{f})}{\mathbb{Q}(\mathbf{y}|\mathbf{f})}\right]^{\Delta\tau(t)}\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1)\,\mathrm{d}\mathbf{f} \\ &\approx \frac{1}{S}\sum_{s=1}^{S}\left[\frac{\mathbb{P}(\mathbf{y}|\mathbf{f}_s)}{\mathbb{Q}(\mathbf{y}|\mathbf{f}_s)}\right]^{\Delta\tau(t)}, \qquad \mathbf{f}_s \sim \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1)\,.\end{aligned}$$

The choice of $\mathbb{Q}(\mathbf{f})$ to be a good approximation to the true posterior makes the fraction $\mathbb{P}(\mathbf{y}|\mathbf{f})/\mathbb{Q}(\mathbf{y}|\mathbf{f})$ as constant as possible, which in turn reduces the error due to the finite step size in thermodynamical integration.

10.3 Algorithm

If only one sample $\mathbf{f}_t$ is used per temperature $\tau(t)$, the value of the entire fraction is obtained as

$$\ln\frac{Z_t}{Z_{t-1}} = \Delta\tau(t)\left[\ln\mathbb{P}(\mathbf{y}|\mathbf{f}_t) - \ln\mathbb{Q}(\mathbf{y}|\mathbf{f}_t)\right]$$

which gives rise to the full estimate

$$\ln Z \approx \sum_{t=1}^{T}\ln\frac{Z_t}{Z_{t-1}} = \ln Z_{\mathbb{Q}} + \sum_{t=1}^{T}\Delta\tau(t)\left[\ln\mathbb{P}(\mathbf{y}|\mathbf{f}_t) + \frac{1}{2}(\mathbf{f}_t-\tilde{\mathbf{m}})^{\top}\mathbf{W}(\mathbf{f}_t-\tilde{\mathbf{m}})\right]$$

for a single run r. The finite temperature change bias can be removed by combining results Z_r from R different runs by their arithmetic mean $\frac{1}{R}\sum_r Z_r$ (Neal, 2001)

$$\ln Z = \ln\int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f} \approx \ln\left(\frac{1}{R}\sum_{r=1}^{R}Z_r\right).$$

Finally, the only primitive needed to obtain MCMC estimates of Z, $\mathbf{m}$ and $\mathbf{V}$ is an efficient sampler for the "intermediate" posterior $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},t-1)$. We use Hybrid Monte Carlo sampling (Neal, 1993).

10.4 Results

If the posterior is very close to the prior (as in regimes 7-9 of Figure 3), it does not make a difference, which we start from. However, if the posterior can be well approximated by a Gaussian

(regimes 4-6), but is sufficiently different from the prior, then the method decreases variance and consequently improves runtimes of AIS. Different approximation methods lead also to differences in the improvement. Namely, the Laplace approximation performs worse than the approximation found by Expectation Propagation because Laplace's method approximates around the mode which can be far away from the mean.

For our evaluations of the approximations to the marginal likelihood, however we started the algorithm from the prior. Otherwise, one might be worried of biasing the MCMC simulation towards the initial distribution in cases where the chain fails to mix properly.

11. Implementation

Implementations of all methods discussed are provided at `http://www.kyb.mpg.de/~hn/approxXX.tar.gz`. The code is designed as an extension to the Gaussian Processes for Machine Learning (GPML) (Rasmussen and Williams, 2006) Matlab Code.[3] Approximate inference for Gaussian processes is done by the `binaryGP.m` function, which takes as arguments the covariance function, the likelihood function and the approximation method. The existing GPML package provides `approxLA.m` for Laplace's method and `approxEP.m` for Expectation Propagation. These implementations are generic to the likelihood function. We provide `cumGauss.m` and `logistic.m` that were designed to avoid numerical problems. In the extension, `approxKL.m`, `approxVB.m`, `approxFV.m` and `approxTAP.m` are included, among others not discussed here, for example sparse and online methods outside the scope of the current investigation. The implementations are straight-forward, although special care has been taken to avoid numerical problems e.g., situations where $\mathbf{K}$ is close to singular. More concretely, we use the well-conditioned matrix[4] $\mathbf{B} = \mathbf{W}^{\frac{1}{2}}\mathbf{K}\mathbf{W}^{\frac{1}{2}} + \mathbf{I} = \mathbf{L}\mathbf{L}^{\top}$ and its Cholesky decomposition to calculate $\mathbf{V} = \left(\mathbf{K}^{-1} + \mathbf{W}\right)^{-1}$ or $\mathbf{k}_*^{\top}\left(\mathbf{K} + \mathbf{W}^{-1}\right)^{-1}\mathbf{k}_*$. The posterior mean is represented in terms of $\boldsymbol{\alpha}$ to avoid multiplications with $\mathbf{K}^{-1}$ and facilitate predictions.

Especially LA and EP show a high level of robustness along the full spectrum of possible hyperparameters. KL uses Gauss-Hermite quadrature; we did not notice problems stemming therefrom. The FV and TAP methods work very reliably, although, we had to add a small (10^{-6}) ridge for FV to regularize $\mathbf{K}$. As a general statement, we did not observe any numerical problems for a wide range of hyperparameters reaching from reasonable values to very extreme scales.

In addition to the code for the algorithms, we provide also a tarball containing all necessary scripts to reproduce the figures of the paper. We offer two versions: The first version contains only the code for running the experiments and drawing the figures.[5] The second version additionally includes the results of the experiments.[6]

12. Experiments

The purpose of the experiments is to illustrate the strengths and weaknesses of the different approximation methods. First of all, the quality of the approximation itself in terms of posterior moments Z,

3. The package is available at `http://www.gaussianprocess.org/gpml/code`.
4. All eigenvalues λ of $\mathbf{B}$ satisfy $1 \leq \lambda \leq 1 + \frac{n}{4}\max_{ij}\mathbf{K}_{ij}$, thus $\mathbf{B}^{-1}$ and $|\mathbf{B}|$ can be safely computed.
5. The code base ($\sim$ 9Mb) can be obtained from `http://www.kyb.mpg.de/~hn/supplement_code.tar.gz`.
6. The complete code base ($\sim$ 400Mb) including all simulation results and scripts to generate figures is stored at `http://www.kyb.mpg.de/~hn/supplement_all.tar.gz`.

m and **V** is studied. At a second level, building on the "low-level" features, we compare predictive performance in terms of the predictive probability p_* given by (Equations 4 and 6):

$$p_* := \mathbb{P}(y_* = 1 | \mathbf{x}_*, \mathbf{y}, \mathbf{X}, \boldsymbol{\theta}) \quad \approx \quad \int \mathrm{sig}(f_*)\, \mathcal{N}\left(f_* | \mu_*, \sigma_*^2\right) \mathrm{d}f_*. \tag{16}$$

On a third level, we assess higher order properties such as the information score, describing how much information the model managed to extract about the target labels, and the error rate—a binary measure of whether a test input is assigned the right class. Uncertainty predictions provided by the model are not captured by the error rate.

Accurate marginal likelihood estimates Z are a key to hyperparameter learning. In that respect, Z can be seen as a high-level feature and as the "zeroth" posterior moment at the same time.

A summary of the whole section is provided in Table 1.

12.1 Data Sets

One main idea of the paper is to study the general behavior of approximate GP classification. Our results for the different approximation methods are not specific to a particular data set but apply to a wide range of application domains. This is reflected by the choice of our reference data sets, widely used in the machine learning literature. Due to limited space, we don't include the full experiments on all data sets in this paper. However, we have verified that the same qualitative conclusions hold for all the data sets considered. The full results are available via the web.[7]

Data set	n_{train}	n_{test}	d	Brief description of problem domain
Breast	300	383	9	Breast cancer[8]
Crabs	100	100	6	Sex of Leptograpsus crabs[9]
Ionosphere	200	151	34	Classification of radar returns from the ionosphere[10]
Pima	350	418	8	Diabetes in Pima Indians[11]
Sonar	108	100	60	Sonar signals bounced by a metal or rock cylinder[12]
USPS 3 vs. 5	767	773	256	Binary sub-problem of the USPS handwritten digit data set[13]

12.2 Results

In the following, we report our experimental results covering posterior moments and predictive performance. Findings for all 5 methods are provided to make the methods as comparable as possible.

7. See links in Footnotes 5 and 6.
8. Data set at `http://mlearn.ics.uci.edu/databases/breast-cancer-wisconsin/`.
9. Data set at `http://www.stats.ox.ac.uk/pub/PRNN/`.
10. Data set at `http://mlearn.ics.uci.edu/databases/ionosphere/`.
11. Data set at `http://mlearn.ics.uci.edu/databases/pima-indians-diabetes/`.
12. Data set at `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/undocumented/connectionist-bench/sonar/`.
13. Data set at `http://www.gaussianprocess.org/gpml/data/`.

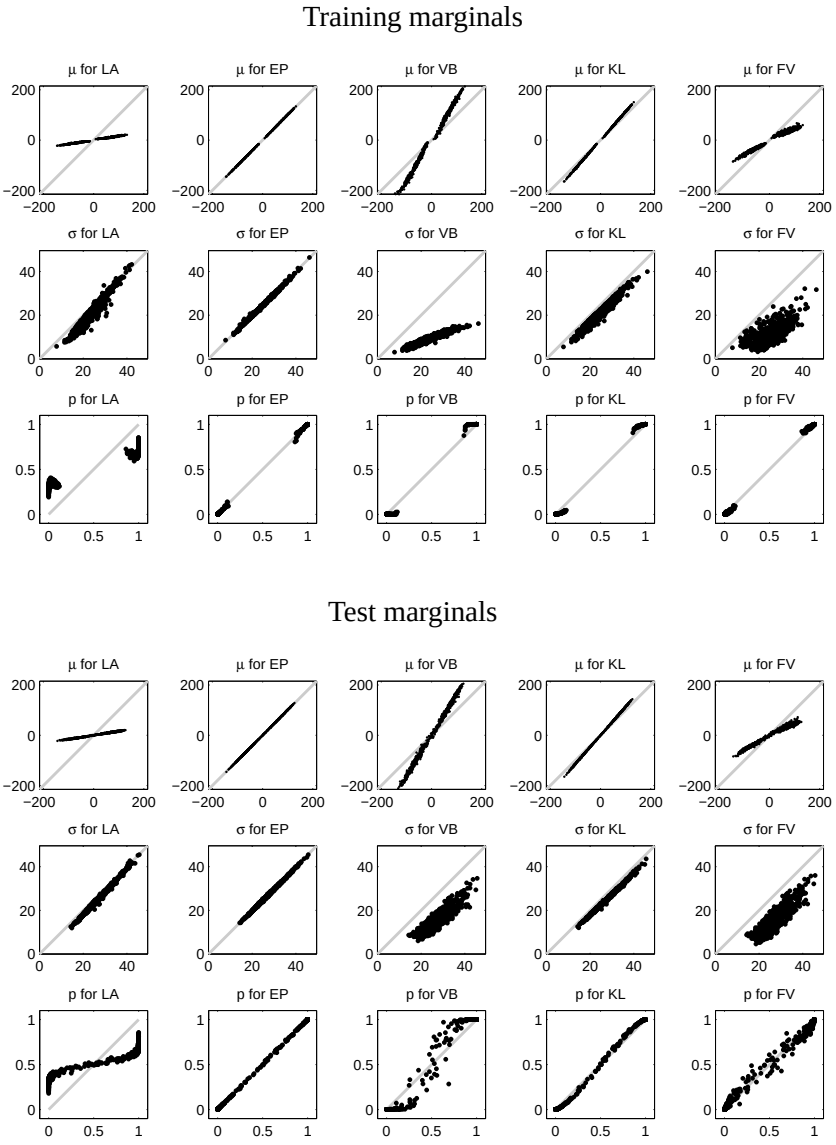


Figure 6: Marginals of USPS 3 vs. 5 for a highly non-Gaussian posterior: Each row consists of five plots showing MCMC ground truth on the x-axis and LA, EP, VB, KL and FV on the y-axis. Based on the logistic likelihood function and the squared exponential covariance function with parameters $\ln \ell = 2.25$ and $\ln \sigma_f = 4.25$ we plot the marginal means, standard deviations and resulting predictive probabilities in rows 1-3. We are working in regime 2 of Figure 3 that means the posterior is highly non-Gaussian. The upper part shows marginals of training points and the lower part shows test point marginals.

	LA	EP*	VB logit|probit	KL	FV	MCMC
idea	quadratic expansion around the mode	marginal moment matching	lower bound on indiv. likelihoods	KL minim., average w.r.t. wrong $\mathbb{Q}(\mathbf{f})$	best free-form factorial	sampling, thermo-dynamic integration
algorithm	Newton steps	iterative matching	Newton steps	Newton steps	fixed-point iteration	Hybrid MC, AIS
complexity	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(8n^3)$	$O(n^3)$	$O(n^3)$
speed	very fast	fast	fast	slow	very fast	very slow
running time	1	10	8	150	4	>500
likelihood properties	1st-3rd log. derivative	$\mathcal{N}$-integrals	lower bound	simple evaluation	$\mathcal{N}$-integrals	1st log derivative
evidence Z	–	$\approx$	– –	–	– – –	=
mean $\mathbf{m}$	– –	$\approx$	++| – –	+	–	=
covariance $\mathbf{V}$	–	$\approx$	– –	–	– –	=
information I	–	$\approx$	$\approx$| –	$\approx$	–	=
PRO	speed	practical accuracy		principled method	speed	theoretical accuracy
CON	mean$\neq$mode, low info I	speed	strong over-confidence	overconfidence	factorizing approxima-tion	very slow

Table 1: Feature summary of the considered algorithms: For each of the six algorithms under consideration, the major properties are listed in the above table. The basic idea of the method along with its computational algorithm and complexity is summarized, the requirements to the likelihood functions are given, the accuracy of evidence and moment estimates as well as information is outlined and some striking advantages and drawbacks are compared. Six relations characterize accuracy: – – – extreme underestimation, – – heavy underestimation, – underestimation, = ground truth, $\approx$ good approximation, + overestimation and ++ heavy overestimation. Running times were calculated by running each algorithm for 9 different hyperparameter regimes and both likelihoods on all data sets. An average running time per data set was calculated for each method and scaled to yield 1 for LA. In the table, the average of these numbers are shown. We are well aware of the fact, that these numbers also depend on our Matlab implementations and choices of convergence thresholds.

12.2.1 MEAN $\mathbf{m}$ AND (CO)VARIANCE $\mathbf{V}$

The posterior process, or equivalently the posterior distribution over the latent values $\mathbf{f}$, is determined by its location parameter $\mathbf{m}$ and its width parameter $\mathbf{V}$. In that respect, these two low-level quantities are the basis for all further calculations. In general, one can say that the methods show

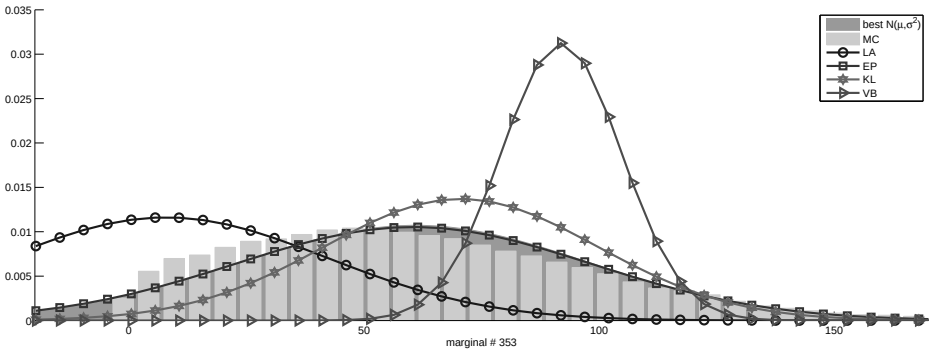


Figure 7: Marginals USPS 3 vs. 5 for digit #353 ≡  : Posterior marginals for one special training point from Figure 6 is shown. Ground truth in terms of true marginal and best Gaussian marginal (matching the moments of the true marginal) are plotted in gray, Gaussian approximations are visualized as lines. For multivariate Gaussians $\mathcal{N}(\mathbf{m},\mathbf{V})$, the *i*-th marginal is given by $\mathcal{N}([\mathbf{m}]_i,[\mathbf{V}]_{ii})$. Thus, the mode m_i of marginal *i* coincides with the *i*-th coordinate of the mode of the joint $[\mathbf{m}]_i$. This relation does not hold for general skewed distribution. Therefore, the marginal given by the Laplace approximation is not centered at the mode of the true marginal.

significant differences in the case of highly non-Gaussian posteriors (regimes 1-5 of Figure 3). Even in the two-dimensional toy example of Figures 4 and 5, significant differences are apparent. The means are inaccurate for LA and VB; whereas the variances are somewhat underestimated by LA and KL and severely so by VB. Marginal means $\mathbf{m}$ and variances $\mathrm{dg}(\mathbf{V})$ for USPS 3 vs. 5 are shown in Figure 6; an exemplary marginal is pictured in Figure 7 for all approximate methods and the MCMC estimate. Along the same lines, a close-to-Gaussian posterior is illustrated in Figure 8. We chose the hyperparameters for the non Gaussian case of Figure 6 to maximize the EP marginal likelihood (see Figure 9), whereas the hyperparameters of Figure 8 were selected to yield a posterior that is almost Gaussian but still has reasonable predictive performance.

The LA method has the principled weakness of expanding around the mode. In high-dimensional spaces, the mode can be very far away from the mean (Kuss and Rasmussen, 2005). The absolute value of the mean is strongly underestimated. Furthermore, the posterior is highly curved at its mode which leads to an underestimated variance, too. These effects can be seen in the first column of Figures 6 and 7, although in the close-to-Gaussian regime LA works well, Figure 8. For large latent function scales σ_f^2, in the limit $\sigma_f^2 \to \infty$, the likelihood becomes a step function, the mode approaches the origin and the curvature at the mode becomes larger. Thus the approximate posterior as found by LA becomes a zero-mean Gaussian which is much too narrow.

The EP method almost perfectly agrees with the MCMC estimates, second column of Figure 6. That means, iterative matching of approximate marginal moments leads to accurate marginal moments of the posterior.

The KL method minimizes the KL-divergence $\mathrm{KL}\left(\mathbb{Q}\left(\mathbf{f}\right) \parallel \mathbb{P}\left(\mathbf{f}\right)\right) = \int \mathbb{Q}\left(\mathbf{f}\right) \ln \frac{\mathbb{Q}(\mathbf{f})}{\mathbb{P}(\mathbf{f})} \mathrm{d}\mathbf{f}$ with the average taken to the approximate distribution $\mathbb{Q}\left(\mathbf{f}\right)$. The method is *zero-forcing* i.e., in regions where $\mathbb{P}\left(\mathbf{f}\right)$ is very small, $\mathbb{Q}\left(\mathbf{f}\right)$ has to be very small as well. In the limit that means $\mathbb{P}\left(\mathbf{f}\right) = 0 \Rightarrow \mathbb{Q}\left(\mathbf{f}\right) = 0$.

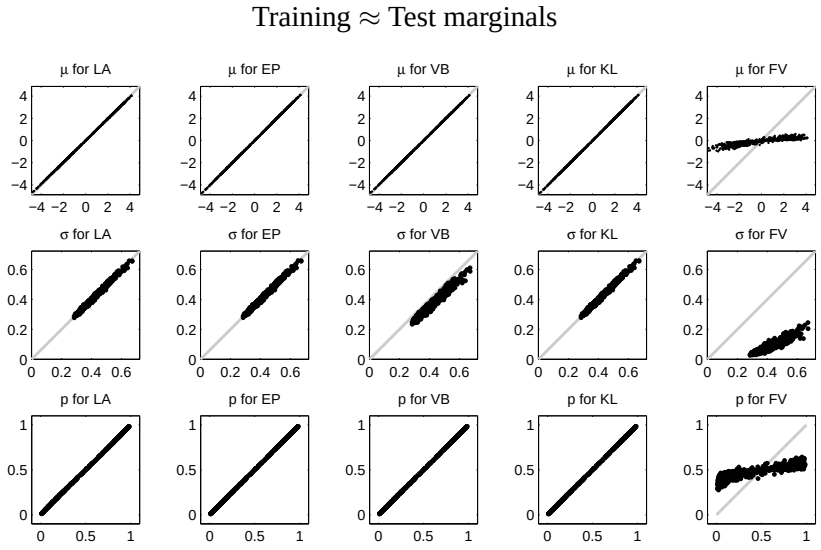


Figure 8: Marginals of USPS 3 vs. 5 for a close-to-Gaussian posterior: Using the squared exponential covariance and the logistic likelihood function with parameters $\ln \ell = 3$ and $\ln \sigma_f = 0.5$, we plot the marginal means, standard deviations and resulting predictive probabilities in rows 1-3. Only the quantities for the trainings set are shown, because the test set results are very similar. We are working in regime 8 of Figure 3 that means the posterior is of rather Gaussian shape. Each row consists of five plots showing MCMC ground truth on the x-axis and LA, EP, VB, KL and FV on the y-axis.

Thus, the support of $\mathbb{Q}(\mathbf{f})$ is smaller than the support of $\mathbb{P}(\mathbf{f})$ and hence the variance is underestimated. Typically, the posterior has a long tail away from zero as seen in Figure 3 regimes 1-5. The zero forcing property shifts the mean of the approximation away from the origin, which results in a slightly overestimated mean, fourth column of Figure 6.

Finally, the VB method can be seen as a more constrained version of the KL method with deteriorated approximation properties. The variance underestimation and mean overestimation is magnified, third column of Figure 6. Due to the required lower bounding property of each individual likelihood term, the approximate posterior has to obey severe restrictions. Especially, the lower bound to the cumulative Gaussian cannot adjust its width since the asymptotic behavior does not depend on the variational parameter (Equation 14).

The FV method has a special rôle because it does not lead to a Gaussian approximation to the posterior but to the closest (in terms of KL-divergence) factorial distribution. If the prior is quite isotropic (regimes 1,4 and 7 of Figure 3), the factorial approximation provides a reasonable approximation. If the latent function values are correlated, the approximation fails. Because of the zero forcing property, mentioned in the discussion of the KL method, both the means and the variances are underestimated. Since a factorial distribution cannot capture correlations, the effect can be severe. It is worth mentioning that there is no difference whether the posterior is close to a

Gaussian or not. In that respect, the FV method complements the LA method, which has difficulties in regimes 1, 2 and 4 of Figure 3.

12.2.2 PREDICTIVE PROBABILITY p_* AND INFORMATION SCORE I

Low-level features like posterior moments are not a goal per se, they are only needed for the purpose of calculating predictive probabilities. Figures 4 and 6 show predictive probabilities in the last row.

In principle, a bad approximation in terms of posterior moments can still provide reasonable predictions. Consider the predictive probability from Equation 16 using a cumulative Gaussian likelihood

$$p_* \quad = \quad \int \mathrm{sig}_{\mathrm{probit}}(f_*)\mathcal{N}(f_*|\mu_*,\sigma_*^2)\mathrm{d}f_* = \mathrm{sig}_{\mathrm{probit}}(\mu_*/\sqrt{1+\sigma_*^2}).$$

It is easy to see that the predictive probability p_* is constant if $\mu_*/\sqrt{1+\sigma_*^2}$ is constant. That means, moving mean μ_* and standard deviation σ_* along the hyperbolic curve $\mu_*^2/C^2-\sigma_*^2=1$, while keeping the sign of μ_* fixed, does not affect the probabilistic prediction. In the limit of large μ_* and large σ_*, rescaling does not change the prediction.

Summarizing all predictive probabilities p_i we consider the scaled information score I. As a baseline model we use the best model ignoring the inputs $\mathbf{x}_i$. This model simply returns predictions matching the class frequencies of the training set

$$B \quad = \quad -\sum_{y=\{+1,-1\}} \frac{n_{\mathrm{test}}^y}{n_{\mathrm{test}}^{+1}+n_{\mathrm{test}}^{-1}} \log_2 \frac{n_{\mathrm{train}}^y}{n_{\mathrm{train}}^{+1}+n_{\mathrm{train}}^{-1}} \leq 1[\mathrm{bit}].$$

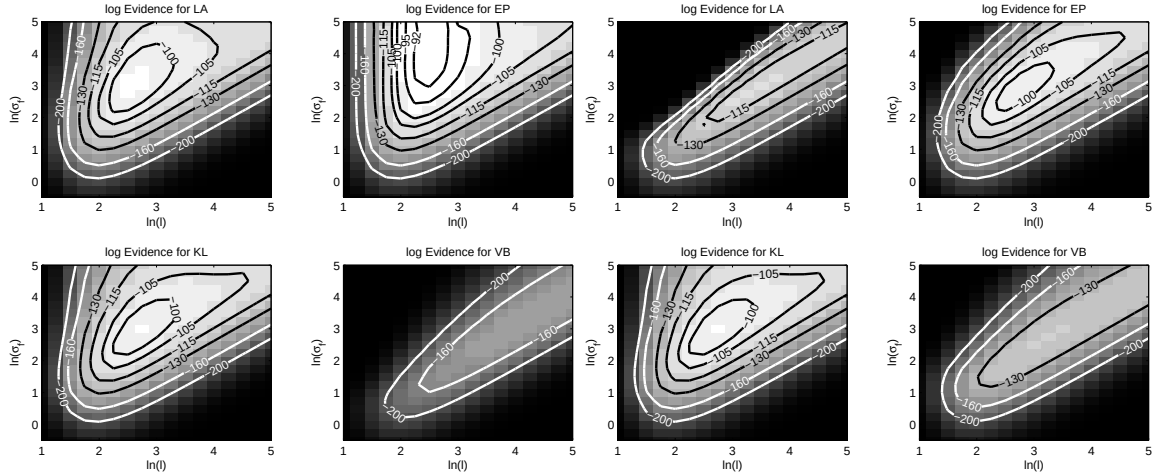
We take the difference between the baseline B (entropy) and the average negative log predictive probabilities $\log_2 \mathbb{P}(y_*|\mathbf{x}_*,\mathbf{y},\mathbf{X})$ to obtain the information score

$$I \quad = \quad B+\frac{1}{2n_{\mathrm{test}}}\sum_{i=1}^{n_{\mathrm{test}}}(1+y_i)\log_2(p_i)+(1-y_i)\log_2(1-p_i),$$

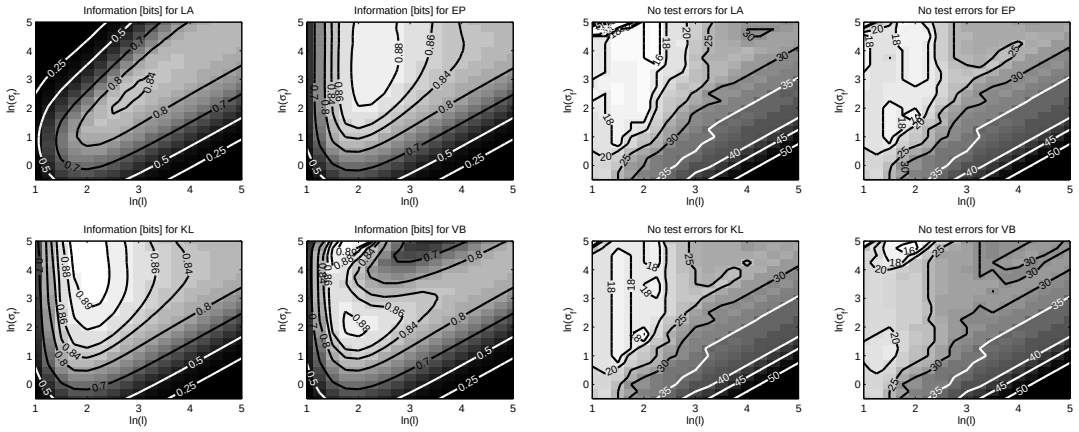
which is $1[\mathrm{bit}]$ for perfect (and confident) prediction and $0[\mathrm{bits}]$ for random guessing (for equiprobable classes). Figures 9(c), 10(middle) and 11(c) contain information scores for 5 different approximation methods on two different data sets as a function of the hyperparameters of the covariance function. According to the EP and KL plots (most prominently in Figure 11(c)), there are two strategies for a model to achieve good predictive performance:

- Find a good length scale ℓ (e.g., $\ln\ell\approx 2$) and choose a latent function scale σ_f above some threshold (e.g., $\ln\sigma_f > 3$).
- Start from a good set of hyperparameters (e.g., $\ln\ell\approx 2$, $\ln\sigma_f\approx 2$) and compensate a harder cutting likelihood ($\sigma_f^2\uparrow$) by making the data points more similar to each other ($\ell^2\uparrow$).

The LA method heavily underestimates the marginal means in the non-Gaussian regime (regimes 1-5 of Figure 3). As a consequence, the predictive probabilities are strongly under-confident in the non-Gaussian regime, first column of Figure 6. The information score's value is too small in the non-Gaussian regime, Figures 9(c) and 11(c).



(a) Evidence (b) Lower bound on evidence



(c) Information in bits (d) Number of errors

Figure 9: Evidence and classification performance for LA, EP, KL & VB on USPS 3 vs. 5: The length scale ℓ and the latent scale σ_f determine the working regime (1-9) of the Gaussian Process as drafted in Figure 3. We use the logistic likelihood and the squared exponential covariance function to classify handwritten digits. The four panels illustrate the model performance in terms of evidence, information and classification errors over the space of hyperparameters (ℓ, σ_f). For better visibility we choose a logarithmic scale of the axes. Panel (a) shows the inherent evidence approximation of the four methods and panel (b) contains the Jensen lower bound (Equation 9) on the evidence used in KL method. Both panels share the same contour levels for all four methods. Note that for the VB method, the general lower bound is a better evidence estimate than the bound provided by the method itself. Panel (c) and (d) show the information score and the number of misclassifications. One can read-off the divergence between posterior and approximation by recalling $\mathrm{KL}(\mathbb{Q}||\mathbb{P}) = \ln Z - \ln Z_B$ from Equation 10 and assuming $\ln Z_{\mathrm{EP}} \approx \ln Z$. In the figure this corresponds to subtracting Subplots (b, LA-VB) from Subplots (a, EP). Obviously, the divergence vanishes for close-to-Gaussian posteriors (regimes 3,5-6,7-9).

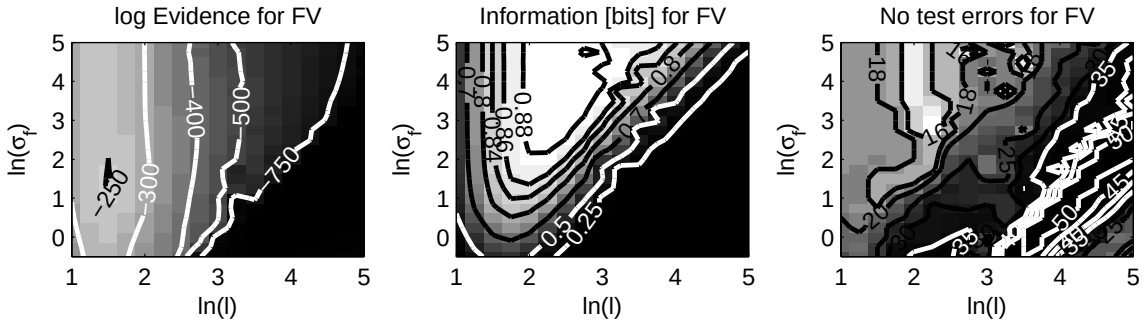


Figure 10: Evidence and classification performance for FV on USPS 3 vs. 5: The plots are a supplement to Figure 9 in that they make the factorial variational method comparable, even though we use the cumulative Gaussian likelihood. The levels of the contour lines for the information score and the number of misclassifications are the same as in Figure 9. For the marginal likelihood other contours are shown, since it has significantly different values.

Since the EP algorithm yields marginal moments very close to the MCMC estimates (second column of Figure 6), its predictive probabilities and information score is consequently also very accurate, Figures 9(c) and 11(c). The plots corresponding to EP can be seen as the quasi gold standard (Kuss and Rasmussen, 2005, Figures 4 and 5).

The KL method slightly underestimates the variance and slightly overestimates the mean which leads to slightly overconfident predictions, fourth column of Figure 6. Overconfidence, in general, leads to a degradation of the information score, however in this example, the information score is very close to the EP values and at the peak it is even slightly (0.01[bits]) higher, Figures 9(c) and 11(c).

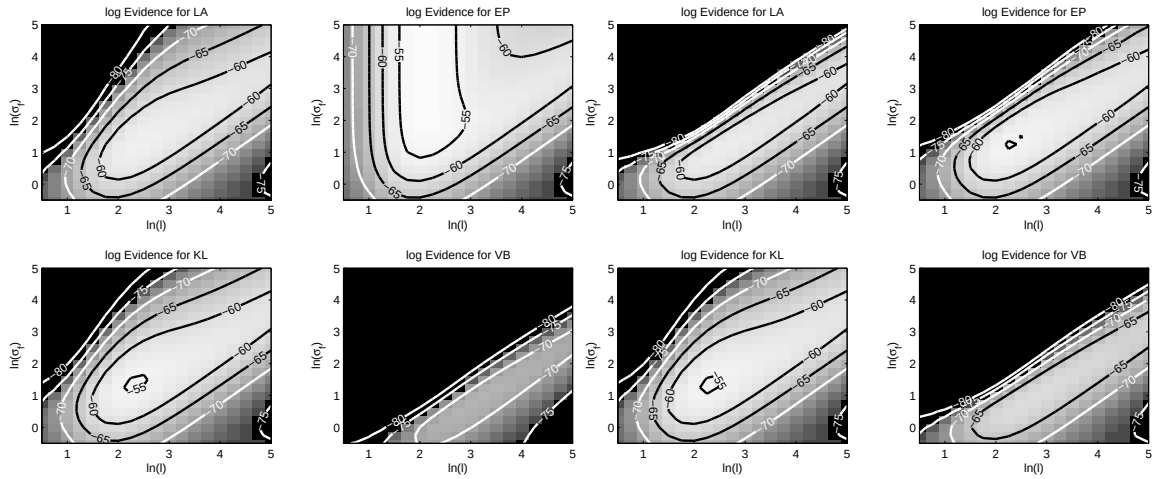
The VB method, again, has the same problems as the KL method only amplified. The predictions are overconfident, third column of Figure 6. Consequently, the information measured score in the non-Gaussian regime is too small. The logistic likelihood function (Figure 9(c)) yields much better results than the cumulative Gaussian likelihood function (Figure 11(c)).

Finally, as the FV method is accurate if the prior is isotropic, predictive probabilities and information scores are very high in regimes 1, 4 and 7 of Figure 3. For correlated priors, the FV method achieves only low information scores, Figure 10(middle). The method seems to benefit from the "hyperbolic scaling invariance" of the predictive probabilities mentioned earlier in that section because both the mean and the variance are strongly underestimated.

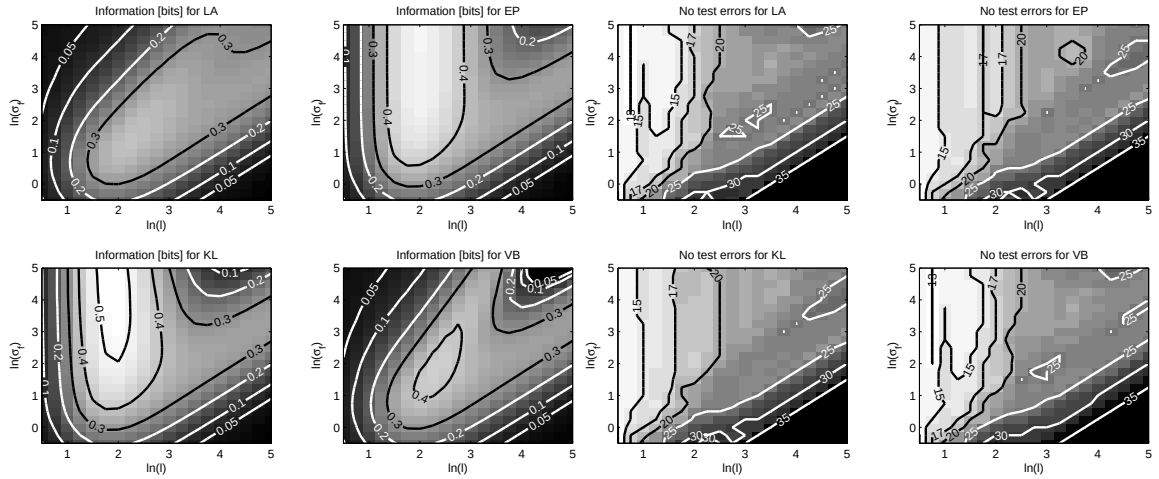
12.2.3 NUMBER OF ERRORS *E*

If one is only interested in the actual class and not in the associated confidence level, one can simply measure the number of misclassifications. Results for 5 approximation methods and 2 data sets are shown in Figures 9(d), 10(right) and 11(d).

Interestingly, all four Gaussian approximation have very similar error rates. The reason is mainly due to the fact that all methods manage to compute the right sign of the marginal mean. Only the FV method with cumulative Gaussian likelihood seems a bit problematic, even though the



(a) Evidence (b) Lower bound on evidence



(c) Information in bits (d) Number of errors

Figure 11: Evidence and classification performance for LA, EP, KL & VB on Sonar: We show the same quantities as in Figure 9, only for the Sonar Mines versus Rocks data set and using the cumulative Gaussian likelihood function.

difference is only very small. Small error rates do not imply high information scores, it is rather the other way round. In Figure 9(d) at $\ln \ell = 2$ and $\ln \sigma_f = 4$ only 16 errors are made by the LA method while the information score (Figure 9(c)) is only of 0.25[bits].

Even the FV method yields very accurate classes, having only small error rates.

12.2.4 MARGINAL LIKELIHOOD Z

Agreement of model and data is typically measured by the marginal likelihood Z. Hyperparameters can conveniently be optimized using Z not least because the gradient $\frac{\partial \ln Z}{\partial \boldsymbol{\theta}}$ can be analytically and efficiently computed for all methods. Formally, the marginal likelihood is the volume of the product of prior and likelihood. In classification, the likelihood is a product of sigmoid functions (Figure 3), so that only the orthant $\{\mathbf{f}|\mathbf{f}\odot\mathbf{y}\geq\mathbf{0}\in\mathbb{R}^n\}$ contains values $\mathbb{P}(\mathbf{f}|\mathbf{y})\geq\frac{1}{2}$. In principle, evidences are bounded by $\ln Z\leq 0$ where $\ln Z=0$ corresponds to a perfect model. As pointed out in Section 2.1.1, the marginal likelihood for a model ignoring the data and having equiprobable targets has the value $\ln Z=-n\ln 2$, which serves as a baseline.

Evidences provided by LA, EP and VB for two data sets are shown in Figures 9(a), 10(left) and 11(a). As the Jensen bound can be applied to any Gaussian approximation of the posterior, we also report it in Figures 9(b) and 11(b).

The LA method strongly underestimates the evidence in the non-Gaussian regime, because it is forced to center its approximation at the mode, Figures 9(a) and 11(a). Nevertheless, there is a good agreement between the value of the marginal likelihood and the corresponding information score. The Jensen lower bound is not tight for the LA approximation, Figures 9(b) and 11(b).

The EP method yields the highest values among all other methods. As described in Section 2.1.2, for high latent function scales σ_f^2, the model becomes effectively independent of σ_f^2. This behavior is only to be seen for the EP method, Figures 9(a) and 11(a). Again, the Jensen bound is not tight for the EP method, Figures 9(b) and 11(b). The difference between EP and MCMC marginal likelihood estimate is vanishingly small (Kuss and Rasmussen, 2005, Figures 4 and 5).

The KL method directly uses the Jensen bound (Equation 8) which can only be tight for Gaussian posterior distributions. If the posterior is very skew, the bound inherently underestimates the marginal likelihood. Therefore, Figures 9(a) and 9(b) and Figures 11(a) and 11(b) show the same values. The disagreement between information score and marginal likelihood makes hyperparameter selection based on the KL method problematic.

The VB method's lower bound on the evidence turns out to be very loose, Figures 9(a) and 11(a). Theoretically, it cannot be better than the more general Jensen bound due to the additional constraints imposed by the individual bound on each likelihood factor, Figures 9(b) and 11(b). In practice, one uses the Jensen bound for hyperparameter selection. Again, the maximum of the bound to the evidence is not very helpful for finding regions of high information score.

Finally, the FV method only yields a poor approximation to the marginal likelihood due to the factorial approximation, Figure 10. The more isotropic the model becomes (small ℓ), the tighter is the bound. For strongly correlated priors (large ℓ) the evidence drops even below the baseline $\ln Z=-n\ln 2$. Thus, the bound is not adequate to do hyperparameter selection as its maximum does not lie in regions with high information score.

12.2.5 CHOICE OF LIKELIHOOD

In the experiments, we worked with two different likelihood functions, namely the logistic and the cumulative Gaussian likelihood. The two functions differ in their slope at the origin and their asymptotic behavior. We did not find empirical evidence supporting the use of either likelihood. Theoretically, the cumulative Gaussian likelihood should be less robust against outliers due to the quadratic asymptotics. Practically, the different slopes result in a shift of the latent function length scale in the order of $\ln\frac{1}{4}-\ln\frac{1}{\sqrt{2\pi}}\approx 0.46$ on a log scale in that the logistic likelihood prefers a

bigger latent scale. Only for the VB method, differences were significant because the logistic bound is more concise. Numerically, however the cumulative Gaussian is preferable.

12.3 Results Across Data Sets

We conclude with a quantitative summary of experiments conducted on 6 data sets (breast, crabs, ionosphere, diabetes, sonar, USPS 3 vs. 5), two different likelihoods (cumulative Gaussian, logistic) and 8 covariance functions (linear, polynomial of degree 1-3, Matérn $\nu \in \{\frac{3}{2}, \frac{5}{2}\}$, squared exponential and neural network) resulting in 96 trials. All 7 approximate classification methods were trained on a 16×16 grid of hyperparameters to compare their behavior under a wide range of conditions. We calculated the maximum (over the hyperparameter grid) amount of information, every algorithm managed to extract from the data in each of the 96 trials. The table shows the number of trials, where the respective algorithm had a maximum information score that was above the mean/median (over the 7 methods).

Test \ Method	**LA**	**EP**	**KL**	**VB**	**FV**	**LR**	**TAPnaive**
# trials, information below **mean**	31	0	0	6	34	92	31
# trials, information below **median**	54	0	0	15	48	96	51

13. Conclusions

In the present paper we provide a comprehensive overview of methods for approximate Gaussian process classification. We present an exhaustive analysis of the considered algorithms using theoretical arguments. We deliver thorough empirical evidence supporting our insights revealing the strengths and weaknesses of the algorithms. Finally, we make a unified and modular implementation of all methods available to the research community.

We are able to conclude that the Expectation Propagation algorithm is, in terms of accuracy, always the method of choice, except when you cannot afford the slightly longer running time compared to the Laplace approximation.

Our comparisons include the Laplace approximation and the Expectation Propagation algorithm (Kuss and Rasmussen, 2005). We extend the latter to the logistic likelihood. We apply Kullback-Leibler divergence minimization to Gaussian process classification and derive an efficient Newton algorithm. Although the principles behind this method have been known for some time, we are unaware that this method has been previously implemented for GPs in practise. The existing variational method (Gibbs and MacKay, 2000) is extended by a lower bound on the cumulative Gaussian likelihood and we provide an implementation based on Newton's method. Furthermore, we give a detailed analysis of the Factorial Variational method (Csató et al., 2000).

All methods are considered in a common framework, approximation quality is assessed, predictive performance is measured and model selection is benchmarked.

In practice, an approximation method has to satisfy a wide range of requirements. If **runtime** is the major concern or one is interested in **error rate** only, the Laplace approximation or label regression should be considered. Only Expectation Propagation and—although a lot slower—the KL-method deliver accurate **marginals** as well as reliable **class probabilities** and allow for faithful **model selection**.

If an application demands a **non-standard likelihood** function, this also affects the choice of the algorithm: The Laplace approximation requires derivatives, Expectation Propagation and the

Factorial Variational method need integrability with respect to Gaussian measures. However, the KL-method simply needs to evaluate the likelihood and known lower bounds naturally lead to the VB algorithm.

Finally, if the classification problem contains a lot of **label noise** (σ_f is small), the exact posterior distribution is effectively close to Gaussian. In that case, the choice of the approximation method is not crucial since in the Gaussian regime, they will give the same answer. For weakly coupled training data, the Factorial Variational method can lead to quite reasonable approximations.

As a future goal remains an in-depth understanding of the properties of sparse and online approximations to the posterior and a coverage of a broader range of covariance functions. Also, the approximation techniques discussed can be applied to other non-Gaussian inference problems besides the narrow applications to binary GP classification discussed here, and there is hope that some of the insights presented may be useful more generally.

Acknowledgments

Thanks to Manfred Opper for pointing us initially to the practical possibility of the KL method and the three anonymous reviewers.

Appendix A. Derivatives

In the following, we provide the expressions for the derivatives needed to implement the VB and the KL method.

A.1 Derivatives for VB

Some notational remarks. Partial derivatives w.r.t. one single parameter such as $\frac{\partial \mathbf{A}_\varsigma}{\partial \varsigma_i}$ or $\frac{\partial \mathbf{b}_\varsigma}{\partial \varsigma_i}$ stay matrices or vectors, respectively. Lowercase letters $\{\mathbf{a},\mathbf{b},\mathbf{c}\}_\varsigma$ indicate vectors, upper case letters $\{\mathbf{A},\mathbf{B},\mathbf{C}\}_\varsigma$ stand for the corresponding diagonal matrices with the vector as diagonal. The dot notation applies to both lower and uppercase letters and denote derivatives w.r.t. the variational parameter vector ς

$$\begin{aligned}
\dot{\mathbf{a}}_\varsigma &:= \left[\frac{\partial a_{\varsigma_i}}{\partial \varsigma_i}\right]_i = \frac{\partial \mathbf{a}_\varsigma}{\partial \varsigma}, \text{ vector,} \\
\ddot{\mathbf{a}}_\varsigma &:= \left[\frac{\partial^2 a_{\varsigma_i}}{\partial \varsigma_i^2}\right]_i = \frac{\partial^2 \mathbf{a}_\varsigma}{\partial \varsigma^2}, \text{ vector,} \\
\dot{\mathbf{A}}_\varsigma &:= \mathrm{Dg}\left(\dot{\mathbf{a}}_\varsigma\right).
\end{aligned}$$

The operators $\mathrm{Dg} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ and $\mathrm{dg} : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ manipulate matrix diagonals. The result of $\mathrm{Dg}(\mathbf{x})$ is a diagonal matrix $\mathbf{X}$ containing $\mathbf{x}$ as diagonal, whereas $\mathrm{dg}(\mathbf{X})$ returns the diagonal of $\mathbf{X}$ as a vector. Hence, we have $\mathrm{Dg}\left(\mathrm{dg}(\mathbf{x})\right) = \mathbf{x}$, but in general $\mathrm{dg}\left(\mathrm{Dg}(\mathbf{X})\right) = \mathbf{X}$ does only hold true for diagonal matrices.

A.1.1 SOME SHORTCUTS USED LATER ONWARDS

$$
\begin{aligned}
\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} &:= \left(\mathbf{K}^{-1} - 2\mathbf{A}_{\boldsymbol{\varsigma}}\right)^{-1} \overset{\mathbf{K} \text{ small}}{=} \mathbf{K} - \mathbf{K}\left(\mathbf{K} - \frac{1}{2}\mathbf{A}_{\boldsymbol{\varsigma}}^{-1}\right)^{-1}\mathbf{K}, \\
\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}} &:= \mathrm{Dg}(\mathbf{y})\mathbf{b}_{\boldsymbol{\varsigma}} = \mathbf{y} \odot \mathbf{b}_{\boldsymbol{\varsigma}}, \\
\mathbf{l}_{\boldsymbol{\varsigma}} &:= \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}} = \left(\mathbf{K}^{-1} - 2\mathbf{A}_{\boldsymbol{\varsigma}}\right)^{-1}(\mathbf{y} \odot \mathbf{b}_{\boldsymbol{\varsigma}}), \\
\frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j} &= \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\left(2\frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j}\mathbf{l}_{\boldsymbol{\varsigma}} + \mathbf{y} \odot \frac{\partial \mathbf{b}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j}\right), \\
\frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \theta_i} &= \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \theta_i}\mathbf{K}^{-1}\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}(\mathbf{y} \odot \mathbf{b}_{\boldsymbol{\varsigma}}), \\
\dot{\mathbf{L}}_{\boldsymbol{\varsigma}} &:= \frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \boldsymbol{\varsigma}^\top} = \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\left(2\mathrm{Dg}(\mathbf{l}_{\boldsymbol{\varsigma}})\dot{\mathbf{A}}_{\boldsymbol{\varsigma}} + \mathrm{Dg}(\mathbf{y})\dot{\mathbf{B}}_{\boldsymbol{\varsigma}}\right), \\
\mathbf{r}_{\boldsymbol{\varsigma}} &:= \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \mathbf{l}_{\boldsymbol{\varsigma}} + \mathrm{dg}\left(\mathbf{l}_{\boldsymbol{\varsigma}}\mathbf{l}_{\boldsymbol{\varsigma}}^\top \dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) \\
&= \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \mathbf{l}_{\boldsymbol{\varsigma}} + \mathbf{l}_{\boldsymbol{\varsigma}} \odot \mathbf{l}_{\boldsymbol{\varsigma}} \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}, \\
\frac{\partial \mathbf{r}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j} &= \mathbf{y} \odot \mathbf{l}_{\boldsymbol{\varsigma}} \odot \frac{\partial \dot{\mathbf{b}}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j} + \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j} + 2\mathbf{l}_{\boldsymbol{\varsigma}} \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}} \odot \frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j} + \mathbf{l}_{\boldsymbol{\varsigma}} \odot \mathbf{l}_{\boldsymbol{\varsigma}} \odot \frac{\partial \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j}, \\
\dot{\mathbf{R}}_{\boldsymbol{\varsigma}} &:= \frac{\partial \mathbf{r}_{\boldsymbol{\varsigma}}}{\partial \boldsymbol{\varsigma}^\top} = \mathrm{Dg}\left(\mathbf{y} \odot \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} + 2\mathbf{l}_{\boldsymbol{\varsigma}} \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right)\dot{\mathbf{L}}_{\boldsymbol{\varsigma}} + \mathrm{Dg}\left(\mathbf{l}_{\boldsymbol{\varsigma}} \odot \left(\mathbf{y} \odot \ddot{\mathbf{b}}_{\boldsymbol{\varsigma}} + \mathbf{l}_{\boldsymbol{\varsigma}} \odot \ddot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right)\right) \\
&= \mathrm{Dg}\left(\mathbf{y} \odot \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} + 2\mathbf{l}_{\boldsymbol{\varsigma}} \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right)\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\mathrm{Dg}\left(\mathbf{y} \odot \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} + 2\mathbf{l}_{\boldsymbol{\varsigma}} \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right) + \mathrm{Dg}\left(\mathbf{l}_{\boldsymbol{\varsigma}} \odot \left(\mathbf{y} \odot \ddot{\mathbf{b}}_{\boldsymbol{\varsigma}} + \mathbf{l}_{\boldsymbol{\varsigma}} \odot \ddot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right)\right).
\end{aligned}
$$

A.1.2 FIRST DERIVATIVES W.R.T. VARIATIONAL PARAMETERS ς_i YIELDING THE GRADIENT

$$
\begin{aligned}
\ln Z_B &= \mathbf{c}_{\boldsymbol{\varsigma}}^\top \mathbb{1} + \frac{1}{2}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}^\top \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}} - \frac{1}{2}\ln\left|\mathbf{I} - 2\mathbf{A}_{\boldsymbol{\varsigma}}\mathbf{K}\right|, \\
\frac{\partial \ln Z_B}{\partial \varsigma_i} &= \frac{\partial c_i}{\partial \varsigma_i} + \tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}^\top \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\left[\mathbf{y} \odot \frac{\partial \mathbf{b}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i} + \frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i}\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}\right] + \mathrm{tr}\left(\left(\mathbf{I} - 2\mathbf{A}_{\boldsymbol{\varsigma}}\mathbf{K}\right)^{-\top}\mathbf{K}\frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i}\right) \\
&\overset{\mathbf{l}_{\boldsymbol{\varsigma}}, \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}}{=} \frac{\partial c_i}{\partial \varsigma_i} + \mathbf{l}_{\boldsymbol{\varsigma}}^\top\left[\mathbf{y} \odot \frac{\partial \mathbf{b}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i} + \frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i}\mathbf{l}_{\boldsymbol{\varsigma}}\right] + \mathrm{tr}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i}\right), \\
\frac{\partial \ln Z_B}{\partial \boldsymbol{\varsigma}} &= \left[\frac{\partial c_i}{\partial \varsigma_i}\right]_i + \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}\right) + \mathrm{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}\tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}^\top\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) + \mathrm{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) \\
&\overset{\mathbf{l}_{\boldsymbol{\varsigma}}}{=} \left[\frac{\partial c_i}{\partial \varsigma_i}\right]_i + \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \mathbf{l}_{\boldsymbol{\varsigma}} + \mathrm{dg}\left(\mathbf{l}_{\boldsymbol{\varsigma}}\mathbf{l}_{\boldsymbol{\varsigma}}^\top\dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) + \mathrm{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) \\
&\overset{\mathbf{r}_{\boldsymbol{\varsigma}}}{=} \left[\frac{\partial c_i}{\partial \varsigma_i}\right]_i + \mathbf{r}_{\boldsymbol{\varsigma}} + \mathrm{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) \\
&= \dot{\mathbf{c}}_{\boldsymbol{\varsigma}} + \mathbf{l}_{\boldsymbol{\varsigma}} \odot \left(\dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} + \mathbf{l}_{\boldsymbol{\varsigma}} \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right) + \mathrm{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\right) \odot \dot{\mathbf{a}}_{\boldsymbol{\varsigma}}.
\end{aligned}
$$

A.1.3 SECOND DERIVATIVES W.R.T. VARIATIONAL PARAMETERS ς_i YIELDING THE HESSIAN

$$\begin{aligned}
\frac{\partial^2 \ln Z_B}{\partial \varsigma_j \partial \varsigma_i} &= \frac{\partial^2 c_i}{\partial \varsigma_j \partial \varsigma_i} + \frac{\partial \mathbf{r}_{\boldsymbol{\varsigma},i}}{\partial \varsigma_j} + \operatorname{tr}\left(2\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j} \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \frac{\partial \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_i} + \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \frac{\partial^2 \mathbf{A}_{\boldsymbol{\varsigma}}}{\partial \varsigma_j \partial \varsigma_i}\right), \\
\frac{\partial^2 \ln Z_B}{\partial \boldsymbol{\varsigma} \partial \boldsymbol{\varsigma}^\top} &= \left[\frac{\partial^2 c_i}{\partial \varsigma_i^2}\right]_{ii} + \frac{\partial \mathbf{r}_{\boldsymbol{\varsigma}}}{\partial \boldsymbol{\varsigma}^\top} + 2\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) \odot \left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right)^\top + \operatorname{Dg}\left(\operatorname{dg}(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}) \odot \ddot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right) \\
&= \ddot{\mathbf{C}}_{\boldsymbol{\varsigma}} + \dot{\mathbf{R}}_{\boldsymbol{\varsigma}} + 2\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right) \odot \left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \dot{\mathbf{A}}_{\boldsymbol{\varsigma}}\right)^\top + \operatorname{Dg}\left(\operatorname{dg}(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}) \odot \ddot{\mathbf{a}}_{\boldsymbol{\varsigma}}\right).
\end{aligned}$$

A.1.4 MIXED DERIVATIVES W.R.T. HYPER- θ_i AND VARIATIONAL PARAMETERS ς_i

$$\begin{aligned}
\frac{\partial^2 \ln Z_B}{\partial \theta_i \partial \boldsymbol{\varsigma}} &= \dot{\mathbf{a}}_{\boldsymbol{\varsigma}} \odot \frac{\partial}{\partial \theta_i}\left(\mathbf{l}_{\boldsymbol{\varsigma}} \odot \mathbf{l}_{\boldsymbol{\varsigma}} + \operatorname{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\right)\right) + \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \theta_i} \\
&= \dot{\mathbf{a}}_{\boldsymbol{\varsigma}} \odot \left(2\mathbf{l}_{\boldsymbol{\varsigma}} \odot \frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \theta_i} + \operatorname{dg}\left(\tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_i} \mathbf{K}^{-1} \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}}\right)\right) + \dot{\mathbf{b}}_{\boldsymbol{\varsigma}} \odot \mathbf{y} \odot \frac{\partial \mathbf{l}_{\boldsymbol{\varsigma}}}{\partial \theta_i}.
\end{aligned}$$

A.1.5 FIRST DERIVATIVES W.R.T. HYPERPARAMETERS θ_i:

For a gradient optimization with respect to $\boldsymbol{\theta}$, we need the gradient of the objective $\partial \ln Z_B / \partial \boldsymbol{\theta}$. Naïvely, the gradient is given by:

$$\begin{aligned}
\frac{\partial \ln Z_B}{\partial \theta_i} &= \frac{1}{2} \tilde{\mathbf{b}}_{\boldsymbol{\varsigma}}^\top \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_i} \mathbf{K}^{-1} \tilde{\mathbf{K}}_{\boldsymbol{\varsigma}} \tilde{\mathbf{b}}_{\boldsymbol{\varsigma}} + \operatorname{tr}\left(\left(\mathbf{I} - 2\mathbf{A}_{\boldsymbol{\varsigma}} \mathbf{K}\right)^{-\top} \mathbf{A}_{\boldsymbol{\varsigma}} \frac{\partial \mathbf{K}}{\partial \theta_i}\right) \\
&\overset{\mathbf{l}_{\boldsymbol{\varsigma}}}{=} \frac{1}{2} \mathbf{l}_{\boldsymbol{\varsigma}}^\top \mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \theta_i} \mathbf{K}^{-1} \mathbf{l}_{\boldsymbol{\varsigma}} + \operatorname{tr}\left(\left(\mathbf{I} - 2\mathbf{A}_{\boldsymbol{\varsigma}} \mathbf{K}\right)^{-\top} \mathbf{A}_{\boldsymbol{\varsigma}} \frac{\partial \mathbf{K}}{\partial \theta_i}\right).
\end{aligned}$$

However, the optimal variational parameter $\boldsymbol{\varsigma}^*$ depends implicitly on the actual choice of $\boldsymbol{\theta}$ and one has to account for that in the derivative by adding an extra "implicit" term

$$\left.\frac{\partial \ln Z_B(\boldsymbol{\theta}, \boldsymbol{\varsigma})}{\partial \theta_i}\right|_{\boldsymbol{\varsigma}=\boldsymbol{\varsigma}^*} = \frac{\partial \ln Z_B(\boldsymbol{\theta}, \boldsymbol{\varsigma}^*)}{\partial \theta_i} + \sum_{j=1}^{n} \frac{\partial \ln Z_B(\boldsymbol{\theta}, \boldsymbol{\varsigma}^*)}{\partial \varsigma_j^*} \frac{\partial \varsigma_j^*}{\partial \theta_i}.$$

The question of how to find an expression for $\frac{\partial \boldsymbol{\varsigma}^*}{\partial \boldsymbol{\theta}}$ can be solved by means of the implicit function theorem for continuous and differentiable functions $\mathbf{F}$:

$$\mathbf{F} : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^n, \quad \mathbf{F}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \quad \Rightarrow \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}}(\mathbf{x}) = -\left(\frac{\partial \mathbf{F}}{\partial \mathbf{y}}(\mathbf{x}, \mathbf{y}(\mathbf{x}))\right)^{-1} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{y}(\mathbf{x})) \text{ if } \mathbf{F}(\mathbf{x}, \mathbf{y}(\mathbf{x})) = \mathbf{0}.$$

Setting $\mathbf{F}(\mathbf{x}, \mathbf{y}) \equiv \frac{\partial \ln Z_B}{\partial \boldsymbol{\varsigma}}(\boldsymbol{\theta}, \boldsymbol{\varsigma})$ leads to

$$\frac{\partial \boldsymbol{\varsigma}_\theta^*}{\partial \boldsymbol{\theta}^\top} = -\left(\frac{\partial^2 \ln Z_B(\boldsymbol{\theta}, \boldsymbol{\varsigma}_\theta^*)}{\partial \boldsymbol{\varsigma} \partial \boldsymbol{\varsigma}^\top}\right)^{-1} \frac{\partial^2 \ln Z_B(\boldsymbol{\theta}, \boldsymbol{\varsigma}_\theta^*)}{\partial \boldsymbol{\theta}^\top \partial \boldsymbol{\varsigma}}$$

and in turn combines to

$$\left.\frac{\partial \ln Z_B}{\partial \theta_i}\right|_{\boldsymbol{\varsigma}=\boldsymbol{\varsigma}^*} = \frac{\partial \ln Z_B}{\partial \theta_i} - \left(\frac{\partial \ln Z_B}{\partial \boldsymbol{\varsigma}}\right)^\top \left(\frac{\partial^2 \ln Z_B}{\partial \boldsymbol{\varsigma} \partial \boldsymbol{\varsigma}^\top}\right)^{-1} \frac{\partial^2 \ln Z_B}{\partial \theta_i \partial \boldsymbol{\varsigma}}$$

where all terms are known.

A.2 Derivatives for KL

The lower bound $\ln Z_B$ to the log marginal likelihood $\ln Z$ is given by Equation 9 as

$$\ln Z \quad \geq \quad = \ln Z_B(\mathbf{m},\mathbf{V}) = a(\mathbf{y},\mathbf{m},\mathbf{V}) + \frac{1}{2}\ln\left|\mathbf{V}\mathbf{K}^{-1}\right| + \frac{n}{2} - \frac{1}{2}\mathbf{m}^\top\mathbf{K}^{-1}\mathbf{m} - \frac{1}{2}\mathrm{tr}\left(\mathbf{V}\mathbf{K}^{-1}\right)$$

where we used the shortcut $a(\mathbf{y},\mathbf{m},\mathbf{V}) = \sum_{i=1}^n \int \mathcal{N}(f_i|m_i,v_{ii}) \ln \mathrm{sig}(y_i f_i) \mathrm{d}f_i$. As a first step, we calculate the first derivatives of $\ln Z_B$ with respect to the posterior moments $\mathbf{m}$ and $\mathbf{V}$ to derive necessary conditions for the optimum by equating them with zero:

$$\begin{aligned}
\frac{\partial \ln Z_B}{\partial \mathbf{V}} = \frac{\partial a(\mathbf{y},\mathbf{m},\mathbf{V})}{\partial \mathbf{V}} + \frac{1}{2}\mathbf{V}^{-1} - \frac{1}{2}\mathbf{K}^{-1} \stackrel{!}{=} \mathbf{0} \quad &\Rightarrow \quad \mathbf{V} = \left(\mathbf{K}^{-1} - 2\mathrm{Dgdg}\frac{\partial a}{\partial \mathbf{V}}\right)^{-1}, \\
\frac{\partial \ln Z_B}{\partial \mathbf{m}} = \frac{\partial a(\mathbf{y},\mathbf{m},\mathbf{V})}{\partial \mathbf{m}} - \mathbf{K}^{-1}\mathbf{m} \stackrel{!}{=} \mathbf{0} \quad &\Rightarrow \quad \mathbf{m} = \mathbf{K}\frac{\partial a}{\partial \mathbf{m}}.
\end{aligned}$$

These two expressions are plugged in the original expression for $\ln Z_B$ using $\mathbf{A} = (\mathbf{I} - 2\mathbf{K}\boldsymbol{\Lambda})^{-1}$ and $\boldsymbol{\Lambda} = \mathrm{Dgdg}\frac{\partial a}{\partial \mathbf{V}}$ to yield:

$$\ln Z_B(\boldsymbol{\alpha},\boldsymbol{\Lambda}) \quad = \quad a\left(\mathbf{y},\mathbf{K}\boldsymbol{\alpha},(\mathbf{K}^{-1}-2\boldsymbol{\Lambda})^{-1}\right) + \frac{1}{2}\ln|\mathbf{A}| - \frac{1}{2}\mathrm{tr}\mathbf{A} + \frac{n}{2} - \frac{1}{2}\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}.$$

Our algorithm uses the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\Lambda}$, so we calculate first and second derivatives to implement Newton's method.

A.2.1 FIRST DERIVATIVES W.R.T. PARAMETERS $\boldsymbol{\alpha}$, $\boldsymbol{\Lambda}$ YIELDING THE GRADIENT

$$\frac{\partial \ln Z_B}{\partial \boldsymbol{\lambda}} = \frac{\partial a}{\partial \boldsymbol{\lambda}} + \mathrm{dg}(\mathbf{V}) - \mathrm{dg}(\mathbf{V}\mathbf{A}^\top) \quad \text{and} \quad \frac{\partial \ln Z_B}{\partial \boldsymbol{\alpha}} = \frac{\partial a}{\partial \boldsymbol{\alpha}} - \mathbf{K}\boldsymbol{\alpha}.$$

Only the terms containing derivatives of a need further attention, namely

$$\frac{\partial a}{\partial \boldsymbol{\alpha}} = \mathbf{K}\frac{\partial a}{\partial \mathbf{m}} \qquad \text{and}$$

$$\begin{aligned}
\mathrm{d}(\mathrm{dg}\mathbf{V}) \quad &= \quad \mathrm{dg}\left[\mathrm{d}\left(\mathbf{K}^{-1}-2\boldsymbol{\Lambda}\right)^{-1}\right] = 2\mathrm{dg}\left[\mathbf{V}\,\mathrm{d}\boldsymbol{\Lambda}\,\mathbf{V}\right] = 2\mathrm{dg}\left[\sum_k \mathbf{v}_k\mathbf{v}_k^\top \mathrm{d}\lambda_k\right] = 2\sum_k (\mathbf{v}_k \odot \mathbf{v}_k)\,\mathrm{d}\lambda_k \\
&= \quad 2\,(\mathbf{V}\odot\mathbf{V})\,\mathrm{d}\boldsymbol{\lambda} \Rightarrow \frac{\partial \mathrm{dg}\mathbf{V}}{\partial \boldsymbol{\lambda}^\top} = 2\mathbf{V}\odot\mathbf{V}, \\
\frac{\partial a}{\partial \boldsymbol{\lambda}} \quad &= \quad 2(\mathbf{V}\odot\mathbf{V})\frac{\partial a(\mathbf{y},\mathbf{m},\mathbf{V})}{\partial \mathrm{dg}\mathbf{V}}.
\end{aligned}$$

As a last step, the derivatives w.r.t. $\mathbf{m}$ and the diagonal part of $\mathbf{V}$ yield

$$\begin{aligned}
\frac{\partial a}{\partial m_i} &= \int \frac{\partial \mathcal{N}(f|m_i,v_{ii})}{\partial m_i} \ln \mathrm{sig}(y_i f)\mathrm{d}f = \int \frac{f-m_i}{v_{ii}} \mathcal{N}(f|m_i,v_{ii}) \ln \mathrm{sig}(y_i f)\mathrm{d}f \\
&= \frac{1}{\sqrt{v_{ii}}} \int f \cdot \mathcal{N}(f) \ln \mathrm{sig}\left(\sqrt{v_{ii}} y_i f + m_i y_i\right) \mathrm{d}f, \\
\frac{\partial a}{\partial v_{ii}} &= \int \frac{\partial \mathcal{N}(f|m_i,v_{ii})}{\partial v_{ii}} \ln \mathrm{sig}(y_i f)\mathrm{d}f = \int \left(\frac{(f-m_i)^2}{v_{ii}^{\frac{3}{2}}} - \frac{1}{\sqrt{v_{ii}}} \right) \mathcal{N}(f|m_i,v_{ii}) \ln \mathrm{sig}(y_i f)\mathrm{d}f \\
&= \frac{1}{2v_{ii}} \int \left(f^2-1\right) \cdot \mathcal{N}(f) \ln \mathrm{sig}\left(\sqrt{v_{ii}} y_i f + m_i y_i\right) \mathrm{d}f.
\end{aligned}$$

A.2.2 SECOND DERIVATIVES W.R.T. PARAMETERS $\boldsymbol{\alpha}$, $\boldsymbol{\Lambda}$ YIELDING THE HESSIAN

Again, we proceed in two steps, calculating derivatives w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\Lambda}$ and by the chain rule compute those w.r.t. $\mathbf{m}$ and $\mathbf{V}$.

$$\begin{aligned}
\frac{\partial^2 \ln Z_B}{\partial \boldsymbol{\alpha} \partial \boldsymbol{\alpha}^\top} &= \frac{\partial^2 a}{\partial \boldsymbol{\alpha} \partial \boldsymbol{\alpha}^\top} + \mathbf{K} = \frac{\partial}{\partial \boldsymbol{\alpha}} \left[\frac{\partial a}{\partial \mathbf{m}^\top} \frac{\partial \mathbf{m}}{\partial \boldsymbol{\alpha}^\top} \right] + \mathbf{K} = \frac{\partial}{\partial \boldsymbol{\alpha}} \left[\frac{\partial a}{\partial \mathbf{m}^\top} \mathbf{K} \right] + \mathbf{K} \\
&= \frac{\partial}{\partial \boldsymbol{\alpha}} \left[\frac{\partial a}{\partial \mathbf{m}^\top} \right] \mathbf{K} + \mathbf{K} = \frac{\partial \mathbf{m}^\top}{\partial \boldsymbol{\alpha}} \frac{\partial}{\partial \mathbf{m}} \left[\frac{\partial a}{\partial \mathbf{m}^\top} \right] \mathbf{K} + \mathbf{K} \\
&= \mathbf{K} \frac{\partial^2 a}{\partial \mathbf{m} \partial \mathbf{m}^\top} \mathbf{K} + \mathbf{K}, \\
\frac{\partial^2 \ln Z_B}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\alpha}^\top} &= \frac{\partial^2 a}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\alpha}^\top} = \frac{\partial}{\partial \boldsymbol{\lambda}} \left[\frac{\partial a}{\partial \mathbf{m}^\top} \right] \mathbf{K} = \frac{\partial \left(\mathrm{dg}\mathbf{V}\right)^\top}{\partial \boldsymbol{\lambda}} \frac{\partial}{\partial \mathrm{dg}\mathbf{V}} \left[\frac{\partial a}{\partial \mathbf{m}^\top} \right] \mathbf{K} \\
&= 2\mathbf{V} \odot \mathbf{V} \frac{\partial^2 a}{\partial \mathrm{dg}\mathbf{V} \partial \mathbf{m}^\top} \mathbf{K}, \\
\frac{\partial^2 \ln Z_B}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\lambda}^\top} &= \frac{\partial^2 a}{\partial \boldsymbol{\lambda} \partial \boldsymbol{\lambda}^\top} + \mathbf{R}, \quad \mathbf{R} := 2\mathbf{V} \odot \left(\mathbf{V} - \mathbf{A}\mathbf{V}^\top - \mathbf{V}\mathbf{A}^\top\right) \\
&= 2 \frac{\partial}{\partial \boldsymbol{\lambda}} \left[\frac{\partial a}{\partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \mathbf{V} \odot \mathbf{V} \right] + \mathbf{R} \\
&= 2 \frac{\partial^2 a}{\partial \boldsymbol{\lambda} \partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \mathbf{V} \odot \mathbf{V} + 2 \left[\frac{\partial a}{\partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \frac{\partial \mathbf{V} \odot \mathbf{V}}{\partial \lambda_i} \right]_i + \mathbf{R} \\
&= 2 \frac{\partial \left(\mathrm{dg}\mathbf{V}\right)^\top}{\partial \boldsymbol{\lambda}} \frac{\partial^2 a}{\partial \mathrm{dg}\mathbf{V} \partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \mathbf{V} \odot \mathbf{V} + 4 \left[\frac{\partial a}{\partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \left(\mathbf{V} \odot \frac{\partial \mathbf{V}}{\partial \lambda_i} \right) \right]_i + \mathbf{R} \\
&= 4\mathbf{V} \odot \mathbf{V} \frac{\partial^2 a}{\partial \mathrm{dg}\mathbf{V} \partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \mathbf{V} \odot \mathbf{V} + 8 \left[\frac{\partial a}{\partial \left(\mathrm{dg}\mathbf{V}\right)^\top} \left(\mathbf{V} \odot \left(\mathbf{v}_i \mathbf{v}_i^\top \right) \right) \right]_i + \mathbf{R}.
\end{aligned}$$

In the following, we abbreviate $\mathcal{N}(f|m_i,v_{ii})$ by $\mathcal{N}_i$.

$$\begin{aligned}
\frac{\partial^2 a}{\partial m_i^2} &= \int \frac{\partial^2 \mathcal{N}_i}{\partial m_i^2} \ln \mathrm{sig}(y_i f) \mathrm{d}f = \int \frac{(f-m_i)^2 - c_{ii}}{v_{ii}^2} \mathcal{N}_i \ln \mathrm{sig}(y_i f) \mathrm{d}f \\
&= \frac{1}{v_{ii}} \int \left(f^2 - 1\right) \cdot \mathcal{N}(f) \ln \mathrm{sig}(\sqrt{v_{ii}} y_i f + m_i y_i) \mathrm{d}f, \\
\frac{\partial^2 a}{\partial c_{ii} \partial m_i} &= \int \frac{\partial^2 \mathcal{N}_i}{\partial v_{ii} \partial m_i} \ln \mathrm{sig}(y_i f) \mathrm{d}f = \int \frac{(f-m_i)^3 - 3(f-m_i) v_{ii}}{2 v_{ii}^3} \mathcal{N}_i \ln \mathrm{sig}(y_i f) \mathrm{d}f \\
&= \frac{1}{2 v_{ii}^{\frac{3}{2}}} \int \left(f^3 - 3f\right) \cdot \mathcal{N}(f) \ln \mathrm{sig}\left(\sqrt{v_{ii}} y_i f + m_i y_i\right) \mathrm{d}f, \\
\frac{\partial^2 a}{\partial v_{ii}^2} &= \int \frac{\partial^2 \mathcal{N}_i}{\partial v_{ii}^2} \ln \mathrm{sig}(y_i f) \mathrm{d}f = \int \frac{(f-m_i)^4 - 6 v_{ii} (f-m_i)^2 + 3 v_{ii}^2}{4 v_{ii}^4} \mathcal{N}_i \ln \mathrm{sig}(y_i f) \mathrm{d}f \\
&= \frac{1}{4 v_{ii}^2} \int \left(f^4 - 6 f^2 + 3\right) \cdot \mathcal{N}(f) \ln \mathrm{sig}(\sqrt{v_{ii}} y_i f + m_i y_i) \mathrm{d}f.
\end{aligned}$$

A.2.3 FIRST DERIVATIVES W.R.T. HYPERPARAMETERS θ_i:

The direct gradient is given by the following equation where we have marked the dependency of the covariance $\mathbf{K}$ on θ_i by subscripts

$$\begin{aligned}
\frac{\partial \ln Z_B(\boldsymbol{\alpha}, \boldsymbol{\Lambda})}{\partial \theta_i} &= \boldsymbol{\alpha}^\top \frac{\partial \mathbf{K}_\theta}{\partial \theta_i} \frac{\partial a(\mathbf{y}, \mathbf{m}, \mathbf{V})}{\partial \mathbf{m}} + \mathrm{dg}\left(\mathbf{A} \frac{\partial \mathbf{K}_\theta}{\partial \theta_i} \mathbf{A}^\top\right)^\top \frac{\partial a(\mathbf{y}, \mathbf{m}, \mathbf{V})}{\partial \mathrm{dg}\mathbf{V}} \\
&\quad + \mathrm{tr}\left(\mathbf{A}^\top \boldsymbol{\Lambda} \frac{\partial \mathbf{K}_\theta}{\partial \theta_i}\right) - \mathrm{tr}\left(\mathbf{A} \frac{\partial \mathbf{K}_\theta}{\partial \theta_i} \boldsymbol{\Lambda} \mathbf{A}\right) - \frac{1}{2} \boldsymbol{\alpha}^\top \frac{\partial \mathbf{K}_\theta}{\partial \theta_i} \boldsymbol{\alpha}.
\end{aligned}$$

Again we have would have to add an implicit term to the gradient, but in our implementation, we forbore from doing so.

Appendix B. Auxiliary Calculations

In the following, we enumerate some calculations we removed from the main text in order to improve on readability.

B.1 Limits of the Covariance Matrix and Corresponding Marginal Likelihood

We investigate the behavior of the covariance matrix $\mathbf{K}$ for extreme lengthscales ℓ. The matrix is given by $[\mathbf{K}]_{ij} = \sigma_f^2 g(|\mathbf{x}_i - \mathbf{x}_j|/\ell)$ where $g : \mathbb{R} \to \mathbb{R}$ is monotonously decreasing and continuous with $g(0) = 1$ and $\lim_{t\to\infty} g(t) = 0$. >From this definition we have $[\mathbf{K}]_{ii} = \sigma_f^2$. We define $\Delta_{ij} := |\mathbf{x}_i - \mathbf{x}_j|/\ell > 0$ for $i \neq j$. From

$$\begin{aligned}
\lim_{\ell\to 0} [\mathbf{K}]_{ij} &\overset{i\neq j}{=} \lim_{\ell\to 0} \sigma_f^2 g(|\mathbf{x}_i - \mathbf{x}_j|/\ell) = \sigma_f^2 \lim_{\Delta_{ij}\to\infty} g(\Delta_{ij}) = 0, \\
\lim_{\ell\to\infty} [\mathbf{K}]_{ij} &\overset{i\neq j}{=} \lim_{\ell\to\infty} \sigma_f^2 g(|\mathbf{x}_i - \mathbf{x}_j|/\ell) = \sigma_f^2 \lim_{\Delta_{ij}\to 0} g(\Delta_{ij}) = 1
\end{aligned}$$

we conclude

$$\begin{aligned}\lim_{\ell\to 0}\mathbf{K} &= \sigma_f^2\mathbf{I},\\ \lim_{\ell\to\infty}\mathbf{K} &= \sigma_f^2\mathbb{1}\mathbb{1}^\top.\end{aligned}$$

The sigmoids are normalized $\mathrm{sig}(-f_i)+\mathrm{sig}(f_i)=1$ and the Gaussian is symmetric $\mathcal{N}(f_i)=\mathcal{N}(-f_i)$. Consequently, we have

$$\begin{aligned}\int \mathrm{sig}(y_i f_i)\,\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i &= \int \mathrm{sig}(f_i)\,\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i\\ &= \int_{-\infty}^{0}\mathrm{sig}(f_i)\,\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i+\int_0^{\infty}\mathrm{sig}(f_i)\,\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i\\ &= \int_0^{\infty}\mathrm{sig}(-f_i)\,\mathcal{N}(-f_i|0,\sigma_f^2)\mathrm{d}f_i+\int_0^{\infty}\mathrm{sig}(f_i)\,\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i\\ &= \int_0^{\infty}[\mathrm{sig}(-f_i)+\mathrm{sig}(f_i)]\,\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i\\ &= \int_0^{\infty}1\cdot\mathcal{N}(f_i|0,\sigma_f^2)\mathrm{d}f_i=\frac{1}{2}.\end{aligned}$$

The marginal likelihood is given by

$$\begin{aligned}Z &= \int \mathbb{P}(\mathbf{y}|\mathbf{f})\,\mathbb{P}(\mathbf{f}|\mathbf{X},\boldsymbol{\theta})\,\mathrm{d}\mathbf{f}\\ &= \int\prod_{i=1}^{n}\mathrm{sig}(y_i f_i)\,|2\pi\mathbf{K}|^{-\frac{1}{2}}\exp(-\frac{1}{2}\mathbf{f}^\top\mathbf{K}^{-1}\mathbf{f})\mathrm{d}\mathbf{f}.\end{aligned}$$

B.1.1 LENGTHSCALE TO ZERO

For $\mathbf{K}=\sigma_f^2\mathbf{I}$ the prior factorizes and we get

$$\begin{aligned}Z_{\ell\to 0} &= \prod_{i=1}^{n}\int \mathrm{sig}(y_i f_i)\,\frac{1}{\sqrt{2\pi\sigma_f^2}}\exp(-\frac{f_i^2}{2\sigma_f^2})\mathrm{d}f_i\\ &\overset{(17)}{=} \prod_{i=1}^{n}\frac{1}{2}=2^{-n}.\end{aligned}$$

B.1.2 LENGTHSCALE TO INFINITY

To get $\mathbf{K} \to \sigma_f^2 \mathbb{1}\mathbb{1}^\top$ we write $\mathbf{K} = \sigma_f^2 \mathbf{1} + \varepsilon^2 \mathbf{I}$ with $\mathbf{1} = \mathbb{1}\mathbb{1}^\top$ and let $\varepsilon \to 0$. The eigenvalue decomposition of $\mathbf{K}$ is written as $\mathbf{K} = \sum_{i=1}^n \mathbf{u}_i \mathbf{u}_i^\top \lambda_i$ with $\mathbf{u}_1 = \frac{1}{\sqrt{n}}\mathbb{1}$, $\lambda_1 = \sigma_f^2 + \varepsilon^2$ and all other $\lambda_i = \varepsilon^2$

$$
\begin{aligned}
Z_{\frac{1}{\varepsilon}} &\overset{\mathbf{K}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top}{=} \int \prod_{i=1}^n \operatorname{sig}(y_i f_i) \, |2\pi\mathbf{\Lambda}|^{-\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{f}^\top \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^\top \mathbf{f}) \mathrm{d}\mathbf{f} \\
&\overset{\mathbf{t}=\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{f}}{=} \int \prod_{i=1}^n \operatorname{sig}\left(y_i \sqrt{\lambda_i} \cdot \mathbf{t}^\top \mathbf{u}_i\right) |2\pi\mathbf{\Lambda}|^{-\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{t}^\top\mathbf{t}) \left|\mathbf{\Lambda}^{\frac{1}{2}}\right| \mathrm{d}\mathbf{t} \\
&= \int \prod_{i=1}^n \operatorname{sig}\left(y_i \sqrt{\lambda_i} \cdot \mathbf{t}^\top \mathbf{u}_i\right) \mathcal{N}(t_i) \mathrm{d}\mathbf{t} \\
&= \int \operatorname{sig}\left(\sqrt{\frac{\sigma_f^2 + \varepsilon^2}{n}} \cdot \mathbf{t}^\top \mathbb{1}\right) \mathcal{N}(t_1) \prod_{i=2}^n \left[\operatorname{sig}\left(\varepsilon \cdot \mathbf{t}^\top \mathbf{u}_i\right)\right] \mathcal{N}(t_i) \mathrm{d}\mathbf{t}, \\
Z_{\ell\to\infty} = \lim_{\varepsilon\to 0} Z &= \int \operatorname{sig}\left(\frac{\sigma_f}{\sqrt{n}} \cdot \mathbf{t}^\top \mathbb{1}\right) \mathcal{N}(t_1) \prod_{i=2}^n \left[\frac{1}{2}\right] \mathcal{N}(t_i) \mathrm{d}\mathbf{t} \\
&\overset{(17)}{=} 2^{-n+1} \int \operatorname{sig}\left(\frac{\sigma_f}{\sqrt{n}} \cdot \mathbf{t}^\top \mathbb{1}\right) \mathcal{N}(\mathbf{t}) \mathrm{d}\mathbf{t} \\
&\overset{r=\mathbf{t}^\top\mathbb{1}}{=} 2^{-n+1} \int \operatorname{sig}\left(\frac{\sigma_f}{\sqrt{n}} \cdot r\right) \mathcal{N}(r) \mathrm{d}r \\
&\overset{(17)}{=} 2^{-n}.
\end{aligned}
$$

B.1.3 LATENT SCALE TO ZERO

We define $\sigma_f^2 \tilde{\mathbf{K}} = \mathbf{K}$ and $\sigma_f \tilde{\mathbf{f}} = \mathbf{f}$ and derive

$$
\begin{aligned}
Z_{\sigma_f} &= \int \prod_{i=1}^n \operatorname{sig}(y_i f_i) \, |2\pi\mathbf{K}|^{-\frac{1}{2}} \exp(-\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}) \mathrm{d}\mathbf{f} \\
&= \int \prod_{i=1}^n \operatorname{sig}\left(y_i \sigma_f \tilde{f}_i\right) |2\pi\mathbf{K}|^{-\frac{1}{2}} \exp(-\frac{\sigma_f^2}{2}\tilde{\mathbf{f}}^\top \mathbf{K}^{-1}\tilde{\mathbf{f}}) \sigma_f^n \mathrm{d}\tilde{\mathbf{f}} \\
&= \int \prod_{i=1}^n \operatorname{sig}\left(y_i \sigma_f \tilde{f}_i\right) \left|2\pi\sigma_f^2\tilde{\mathbf{K}}\right|^{-\frac{1}{2}} \exp(-\frac{\sigma_f^2}{2}\tilde{\mathbf{f}}^\top \sigma_f^{-2}\tilde{\mathbf{K}}^{-1}\tilde{\mathbf{f}}) \sigma_f^n \mathrm{d}\tilde{\mathbf{f}} \\
&= \int \prod_{i=1}^n \left[\operatorname{sig}\left(y_i \sigma_f \tilde{f}_i\right)\right] \mathcal{N}\left(\tilde{\mathbf{f}} | \mathbf{0}, \tilde{\mathbf{K}}\right) \mathrm{d}\tilde{\mathbf{f}}, \\
Z_{\sigma_f\to 0} = \lim_{\sigma_f\to 0} Z &= \int \prod_{i=1}^n \left[\frac{1}{2}\right] \mathcal{N}\left(\tilde{\mathbf{f}} | \mathbf{0}, \tilde{\mathbf{K}}\right) \mathrm{d}\tilde{\mathbf{f}} = 2^{-n}.
\end{aligned}
$$

Note that the functions, we are using are all well-behaved, such that the limits do exist.

B.2 Posterior Divided by Prior = Effective Likelihood

$$\begin{aligned}
\mathbb{Q}(\mathbf{y}|\mathbf{f}) &= \frac{\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})}{\mathbb{P}(\mathbf{f}|\mathbf{X})} = \frac{\mathcal{N}\left(\mathbf{f}|\mathbf{m},\left(\mathbf{K}^{-1}+\mathbf{W}\right)^{-1}\right)}{\mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})} \\
&= \frac{\mathcal{N}\left(\mathbf{f}|\tilde{\mathbf{m}},\mathbf{W}^{-1}\right)}{\mathcal{N}(\tilde{\mathbf{m}}|\mathbf{0},\mathbf{K}+\mathbf{W}^{-1})}, \quad \tilde{\mathbf{m}} = (\mathbf{K}\mathbf{W})^{-1}\mathbf{m}+\mathbf{m} \\
&= \frac{(2\pi)^{-\frac{n}{2}}\left|\mathbf{W}^{-1}\right|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{f}-\tilde{\mathbf{m}})^{\top}\mathbf{W}(\mathbf{f}-\tilde{\mathbf{m}})\right)}{(2\pi)^{-\frac{n}{2}}\left|\mathbf{K}+\mathbf{W}^{-1}\right|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\tilde{\mathbf{m}}^{\top}(\mathbf{K}+\mathbf{W}^{-1})^{-1}\tilde{\mathbf{m}}\right)} \\
&= \sqrt{|\mathbf{K}\mathbf{W}+\mathbf{I}|}\frac{\exp\left(-\frac{1}{2}(\mathbf{f}-\tilde{\mathbf{m}})^{\top}\mathbf{W}(\mathbf{f}-\tilde{\mathbf{m}})\right)}{\exp\left(-\frac{1}{2}\tilde{\mathbf{m}}^{\top}(\mathbf{K}+\mathbf{W}^{-1})^{-1}\tilde{\mathbf{m}}\right)} \\
&=: \frac{1}{Z_{\mathbb{Q}}}\exp\left(-\frac{1}{2}(\mathbf{f}-\tilde{\mathbf{m}})^{\top}\mathbf{W}(\mathbf{f}-\tilde{\mathbf{m}})\right), \\
\ln Z_{\mathbb{Q}} &= -\frac{1}{2}\tilde{\mathbf{m}}^{\top}\left(\mathbf{K}+\mathbf{W}^{-1}\right)^{-1}\tilde{\mathbf{m}}-\frac{1}{2}\ln|\mathbf{K}\mathbf{W}+\mathbf{I}|
\end{aligned}$$

B.3 Kullback-Leibler Divergence for KL method

We wish to calculate the divergence between the approximate posterior, a Gaussian, and the true posterior

$$\begin{aligned}
\mathrm{KL}\left(\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) \parallel \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\right) &= \int \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\ln\frac{\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})}{\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})}\mathrm{d}\mathbf{f} \\
&\overset{(2)}{=} \int \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\ln\frac{Z\cdot\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})}{\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\prod_{i=1}^{n}\mathbb{P}(y_i|f_i)}\mathrm{d}\mathbf{f} \\
&= \ln Z + \int \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\ln\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\,\mathrm{d}\mathbf{f} \\
&\quad - \int \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\ln\prod_{i=1}^{n}\mathbb{P}(y_i|f_i)\mathrm{d}\mathbf{f} \\
&\quad - \int \mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V})\ln\mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})\mathrm{d}\mathbf{f}.
\end{aligned}$$

There are three Gaussian integrals to evaluate; the entropy of the approximate posterior and two other expectations

$$\begin{aligned}
\mathrm{KL}\left(\mathbb{Q}(\mathbf{f}|\boldsymbol{\theta}) \parallel \mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X},\boldsymbol{\theta})\right) &= \ln Z - \frac{1}{2}\ln|\mathbf{V}| - \frac{n}{2} - \frac{n}{2}\ln 2\pi \\
&\quad - \int \mathcal{N}(f)\left[\sum_{i=1}^{n}\ln\mathrm{sig}\left(\sqrt{v_{ii}}y_i f + m_i y_i\right)\right]\mathrm{d}f \qquad (17) \\
&\quad + \frac{n}{2}\ln 2\pi + \frac{1}{2}\ln|\mathbf{K}| + \frac{1}{2}\mathbf{m}^{\top}\mathbf{K}^{-1}\mathbf{m} + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{V}\right).
\end{aligned}$$

Summing up and dropping the constant (w.r.t. $\mathbf{m}$ and $\mathbf{V}$) terms, we arrive at

$$\mathrm{KL}(\mathbf{m},\mathbf{V}) \stackrel{c}{=} -\int \mathcal{N}(f)\left[\sum_{i=1}^{n} \ln \mathrm{sig}\left(\sqrt{v_{ii}}y_i f + m_i y_i\right)\right] \mathrm{d}f - \frac{1}{2}\ln|\mathbf{V}| + \frac{1}{2}\mathbf{m}^\top \mathbf{K}^{-1}\mathbf{m} + \frac{1}{2}\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{V}\right).$$

B.4 Gaussian Integral for VB Lower Bound

$$\begin{aligned}
Z_B &= \int \mathbb{P}(\mathbf{f}|\mathbf{X})\,\mathbb{Q}(\mathbf{y}|\mathbf{f},\mathbf{A},\mathbf{b},\mathbf{c})\,\mathrm{d}\mathbf{f} = \int \mathcal{N}(\mathbf{f}|0,\mathbf{K}) \exp\left(\mathbf{f}^\top \mathbf{A}\mathbf{f} + (\mathbf{b}\odot\mathbf{y})^\top \mathbf{f} + \mathbf{c}^\top \mathbb{1}\right) \mathrm{d}\mathbf{f} \\
&= \frac{\exp\left(\mathbf{c}^\top \mathbb{1}\right)}{\sqrt{(2\pi)^n |\mathbf{K}|}} \int \exp\left(-\frac{1}{2}\mathbf{f}^\top \left(\mathbf{K}^{-1} - 2\mathbf{A}\right)\mathbf{f} + (\mathbf{b}\odot\mathbf{y})^\top \mathbf{f}\right) \mathrm{d}\mathbf{f} \\
&= \frac{\exp\left(\mathbf{c}^\top \mathbb{1}\right)}{\sqrt{(2\pi)^n |\mathbf{K}|}} \sqrt{\frac{(2\pi)^n}{|\mathbf{K}^{-1} - 2\mathbf{A}|}} \exp\left(\frac{1}{2}(\mathbf{b}\odot\mathbf{y})^\top \left(\mathbf{K}^{-1} - 2\mathbf{A}\right)^{-1} (\mathbf{b}\odot\mathbf{y})\right) \\
&= \frac{\exp\left(\mathbf{c}^\top \mathbb{1}\right)}{\sqrt{|\mathbf{I} - 2\mathbf{A}\mathbf{K}|}} \exp\left(\frac{1}{2}(\mathbf{b}\odot\mathbf{y})^\top \left(\mathbf{K}^{-1} - 2\mathbf{A}\right)^{-1} (\mathbf{b}\odot\mathbf{y})\right), \\
\ln Z_B &= \mathbf{c}^\top \mathbb{1} + \frac{1}{2}(\mathbf{b}\odot\mathbf{y})^\top \left(\mathbf{K}^{-1} - 2\mathbf{A}\right)^{-1} (\mathbf{b}\odot\mathbf{y}) - \frac{1}{2}\ln|\mathbf{I} - 2\mathbf{A}\mathbf{K}|.
\end{aligned}$$

B.5 Lower Bound for the Cumulative Gaussian Likelihood

A lower bound

$$\mathrm{sig}_{\mathrm{probit}}(y_i f_i) \;\geq\; \mathbb{Q}(y_i|f_i,\varsigma_i) = a_i f_i^2 + b_i f_i + c_i$$

for the cumulative Gaussian likelihood function is derived by matching the function at one point ς

$$\mathbb{Q}(y_i = +1|f_i,\varsigma_i) \;=\; \mathrm{sig}_{\mathrm{probit}}(\varsigma_i), \forall i$$

and by matching the first derivative

$$\left.\frac{\partial}{\partial f_i}\ln \mathbb{Q}(y_i = +1|f_i,\varsigma_i)\right|_{\varsigma_i} \;=\; \frac{\partial \ln \mathrm{sig}_{\mathrm{probit}}(y_i f_i)}{\partial f_i} = \frac{\mathcal{N}(\varsigma_i)}{\mathrm{sig}_{\mathrm{probit}}(\varsigma_i)}, \forall i$$

at this point for a tight approximation. Solving for these constraints leads to the coefficients

$$\begin{aligned}
\text{asymptotic behavior} \Rightarrow a_i &= -\frac{1}{2}, \\
\text{first derivative} \Rightarrow b_i &= \varsigma_i + \frac{\mathcal{N}(\varsigma_i)}{\mathrm{sig}_{\mathrm{probit}}(\varsigma_i)}, \\
\text{point matching} \Rightarrow c_i &= \left(\frac{\varsigma_i}{2} - b_i\right)\varsigma_i + \log \mathrm{sig}_{\mathrm{probit}}(\varsigma_i).
\end{aligned}$$

B.6 Free Form Optimization for FV

We make a factorial approximation $\mathbb{P}(\mathbf{f}|\mathbf{y},\mathbf{X}) \approx \mathbb{Q}(\mathbf{f}) := \prod_i \mathbb{Q}(f_i)$ to the posterior by minimizing

$$\begin{aligned}\mathrm{KL}[\mathbb{Q}(\mathbf{f})||\mathbb{P}(\mathbf{f})] &= \int \prod_{i=1}^n \mathbb{Q}(f_i) \ln \frac{Z \cdot \prod_{i=1}^n \mathbb{Q}(f_i)}{\mathcal{N}(\mathbf{f}|\mathbf{m},\mathbf{V}) \prod_{i=1}^n \mathbb{P}(y_i|f_i)} \mathrm{d}\mathbf{f} \\ &= \sum_i \int \mathbb{Q}(f_i) \ln \frac{\mathbb{Q}(f_i)}{\mathbb{P}(y_i|f_i)} \mathrm{d}f_i + \frac{1}{2} \int \prod_{i=1}^n \mathbb{Q}(f_i) \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f} \mathrm{d}\mathbf{f} + \mathrm{const}_{\mathbf{f}}.\end{aligned}$$

Free-form optimization proceeds by equating the functional derivative with zero

$$\frac{\delta \mathrm{KL}}{\delta \mathbb{Q}(f_i)} = \ln \mathbb{Q}(f_i) + 1 - \ln \mathbb{P}(y_i|f_i) + \frac{1}{2} \frac{\delta}{\delta \mathbb{Q}(f_i)} \int \prod_{i=1}^n \mathbb{Q}(f_i) \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f} \mathrm{d}\mathbf{f}. \tag{18}$$

We abbreviate the integral in the last term with ξ and rewrite it in terms of simple one-dimensional integrals $m_l = \int f_l \mathbb{Q}(f_l) \mathrm{d}f_l$ and $v_l = \int f_l^2 \mathbb{Q}(f_l) \mathrm{d}f_l - m_l^2$

$$\begin{aligned}\xi &= \int \prod_i \mathbb{Q}_i \sum_{j,k} f_j \left[\mathbf{K}^{-1}\right]_{jk} f_k \mathrm{d}\mathbf{f} \\ &= \int \prod_{i \neq l} \mathbb{Q}_i \left[\int \mathbb{Q}_l \left(f_l^2 \left[\mathbf{K}^{-1}\right]_{ll} + 2 f_l \sum_{j \neq l} f_j \left[\mathbf{K}^{-1}\right]_{jl} + \sum_{j \neq l, k \neq l} f_j \left[\mathbf{K}^{-1}\right]_{jk} f_k \right) \mathrm{d}f_l \right] \mathrm{d}\mathbf{f}_{\neg l} \\ &= \int \prod_{i \neq l} \mathbb{Q}_i \left[\left[\mathbf{K}^{-1}\right]_{ll} \underbrace{\int f_l^2 \mathbb{Q}_l \mathrm{d}f_l}_{v_l + m_l^2} + 2 \left(\sum_{j \neq l} f_j \left[\mathbf{K}^{-1}\right]_{jl}\right) \underbrace{\int f_l \mathbb{Q}_l \mathrm{d}f_l}_{m_l} + \sum_{j \neq l, k \neq l} f_j \left[\mathbf{K}^{-1}\right]_{jk} f_k \right] \mathrm{d}\mathbf{f}_{\neg l} \\ &= \left[\mathbf{K}^{-1}\right]_{ll} (v_l + m_l^2) + 2 \sum_{j \neq l} m_j \left[\mathbf{K}^{-1}\right]_{jl} m_l + \int \prod_{i \neq l} \mathbb{Q}_i \sum_{j \neq l, k \neq l} f_j \left[\mathbf{K}^{-1}\right]_{jk} f_k \mathrm{d}\mathbf{f}_{\neg l} \\ &= \text{induction over } l \\ &= \sum_l \left[\mathbf{K}^{-1}\right]_{ll} (v_l + m_l^2) + 2 \sum_{j<l} m_j \left[\mathbf{K}^{-1}\right]_{jl} m_l.\end{aligned}$$

Plugging this into Equation 18 and using $\frac{\delta \int f_l^p \mathbb{Q}(f_l) \mathrm{d}f_l}{\delta \mathbb{Q}(f_l)} = f_l^p$, we find

$$\begin{aligned}\frac{\delta \mathrm{KL}}{\delta \mathbb{Q}(f_i)} &= \ln \mathbb{Q}(f_i) + 1 - \ln \mathbb{P}(y_i|f_i) + \frac{1}{2} f_i \left[\mathbf{K}^{-1}\right]_{ii} f_i + f_i \sum_l \left[\mathbf{K}^{-1}\right]_{il} m_l \stackrel{!}{\equiv} 0 \\ \Rightarrow \mathbb{Q}(f_i) &\propto \exp\left(-\frac{1}{2} f_i \left[\mathbf{K}^{-1}\right]_{ii} f_i - f_i \sum_{l \neq i} \left[\mathbf{K}^{-1}\right]_{il} m_l \right) \mathbb{P}(y_i|f_i) \\ \Rightarrow \mathbb{Q}(f_i) &\propto \mathcal{N}\left(f_i \middle| m_i - \frac{\left[\mathbf{K}^{-1}\mathbf{m}\right]_i}{\left[\mathbf{K}^{-1}\right]_{ii}}, \left[\mathbf{K}^{-1}\right]_{ii}^{-1} \right) \mathbb{P}(y_i|f_i)\end{aligned}$$

as the functional form of the best possible factorial approximation, namely a product of the true likelihood times a Gaussian with the same precision as the prior marginal.

References

Yasemin Altun, Thomas Hofmann, and Alex Smola. Gaussian process classification for segmenting and annotating sequences. In *International Conference on Machine Learning*, 2004.

Wei Chu, Zoubin Ghahramani, Francesco Falciani, and David L. Wild. Biomarker discovery in microarray gene expression data with gaussian processes. *Bioinformatics*, 21:3385–3393, 2005.

Lehel Csató, Ernest Fokoué, Manfred Opper, and Bernhard Schottky. Efficient Approaches to Gaussian Process Classification. In *Neural Information Processing Systems 12*, pages 251–257. MIT Press, 2000.

Mark N. Gibbs and David J. C. MacKay. Variational Gaussian Process Classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464, 2000.

Mark Girolami and Simon Rogers. Variational Bayesian Multinomial Probit Regression with Gaussian Process Priors. *Neural Computation*, 18:1790–1817, 2006.

Ashish Kapoor and Rosalind W. Picard. Multimodal affect recognition in learning environments. In *ACM international conference on Multimedia*, 2005.

Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007.

Malte Kuss and Carl Edward Rasmussen. Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 6:1679 – 1704, 10 2005.

David J. C. MacKay. Bayesian Interpolation. *Neural Computation*, 4(3):415–447, 1992.

Thomas P. Minka. Expectation Propagation for Approximate Bayesian Inference. In *UAI*, pages 362–369. Morgan Kaufmann, 2001a.

Thomas P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2001b.

Tom Minka. Divergence Measures and Message Passing. Technical report, Microsoft Research, 2005.

Radford M. Neal. Annealed Importance Sampling. *Statistics and Computing*, 11:125–139, 2001.

Radford M. Neal. Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, September 1993.

Manfred Opper and Cédric Archambeau. The Variational Gaussian Approximation Revisited. *Neural Computation*, accepted, 2008.

Manfred Opper and Ole Winther. Gaussian Processes for Classification: Mean Field Algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

Manfred Opper and Ole Winther. Expectation Consistent Approximate Inference. *Journal of Machine Learning Research*, 6:2177–2204, 2005.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, February 1993.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *JMLR*, 5:101–141, 2004.

Anton Schwaighofer, Volker Tresp, Peter Mayer, Alexander K. Scheel, and Gerhard Müller. The RA scanner: Prediction of rheumatoid joint inflammation based on laser imaging. In *NIPS*, 2002.

Matthias Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.

Matthias Seeger. Bayesian Methods for Support Vector Machines and Gaussian Processes. Master's thesis, Universität Karlsruhe, 1999.

S. Sundararajan and S. S. Keerthi. Predictive Approaches for Choosing Hyperparameters in Gaussian Processes. *Neural Computation*, 13:1103–1118, 2001.

Christopher K. I. Williams and David Barber. Bayesian Classification with Gaussian Processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(20):1342–1351, 1998.

Mingjun Zhong, Fabien Lotte, Mark Girolami, and Anatole Lécuyer. Classifying eeg for brain computer interfaces using gaussian processes. *Pattern Recognition Letters*, 29:354–359, 2008.

Journal of Machine Learning Research 9 (2008) 2079-2111 Submitted 8/08; Published 10/08

Value Function Approximation using Multiple Aggregation for Multiattribute Resource Management

Abraham George AGEORGE@PRINCETON.EDU
Warren B. Powell POWELL@PRINCETON.EDU
Department of Operations Research and Financial Engineering
Princeton University
Princeton, NJ 08544, USA

Sanjeev R. Kulkarni KULKARNI@PRINCETON.EDU
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544, USA

Editor: Sridhar Mahadevan

Abstract

We consider the problem of estimating the value of a multiattribute resource, where the attributes are categorical or discrete in nature and the number of potential attribute vectors is very large. The problem arises in approximate dynamic programming when we need to estimate the value of a multiattribute resource from estimates based on Monte-Carlo simulation. These problems have been traditionally solved using aggregation, but choosing the right level of aggregation requires resolving the classic tradeoff between aggregation error and sampling error. We propose a method that estimates the value of a resource at different levels of aggregation simultaneously, and then uses a weighted combination of the estimates. Using the optimal weights, which minimizes the variance of the estimate while accounting for correlations between the estimates, is computationally too expensive for practical applications. We have found that a simple inverse variance formula (adjusted for bias), which effectively assumes the estimates are independent, produces near-optimal estimates. We use the setting of two levels of aggregation to explain why this approximation works so well.

Keywords: hierarchical statistics, approximate dynamic programming, mixture models, adaptive learning, multiattribute resources

1. Introduction

We consider the problem of managing resources (people, equipment) that can be described using a vector of attributes $a = (a_1, a_2, \ldots, a_M)$. Our work has grown out of a series of projects with industry and the military that involve managing resources over time under uncertainty. In all of these projects, we use algorithms that require estimating the marginal value of a resource with attribute vector a. As these projects have made the transition from laboratory experiments to industrial implementations, we have found that one and two dimensional attributes (for example, location and possibly equipment type) quickly grow to five or ten dimensions, with an exponential growth in the number of potential attributes. Examples of actual projects we have worked on which exhibit this behavior include:

- Managing pilots for business jets - The attributes of a pilot include elements such as home city, number of days away from home and the equipment that he is trained to fly. Decisions about pilots can include assigning a pilot to a particular flight, or a decision to send a pilot for training on a new type of aircraft.

- Managing locomotives - The decision to assign a particular locomotive to a particular train has to consider attributes such as the type of locomotive, the number of days until it has to be maintained, its current location and its home maintenance shop.

- Managing a fleet of freight cars - Freight cars have attributes such as location, time until arrival to a destination, loaded or empty status, ownership, and maintenance status.

- Managing a fleet of trucks to move loads - The truck can be described using attributes such as its current location, the home domicile of the driver, the maintenance level and whether it is being driven by a solo driver or a team of two drivers. Decisions include where to move to and whether to move loaded or empty.

- Managing cargo aircraft for the military - We have to decide which aircraft should be assigned to satisfy a particular requirement (a movement of freight or passengers). Choosing the best aircraft requires knowing the value of an aircraft at the destination which depends on the type of aircraft, cargo configuration, whether it is loaded or empty (and if loaded, the load characteristics), and its maintenance status.

- Managing blood inventories - Blood is characterized by blood type, age, location, and whether it has been frozen. New supplies of, and the demand for, blood is random.

All of these are examples of resource allocation problems where a decision has to be made now to act on resources (trucks, jets, locomotives) which will bring about a change in their attributes. Let $a \in \mathcal{A}$ be the attribute vector describing a resource now. If we act on the resource, we may produce a resource with attribute a' with value $v_{a'}$. In a dynamic programming setting, the value $v_{a'}$ refers to the solution of a finite horizon discounted reward dynamic program. In practical applications, we cannot compute $v_{a'}$ exactly, so we resort to Monte Carlo methods where we might observe random observations $\hat{v}_{a'}$ and use these to produce a statistical estimate $\bar{v}_{a'}$ (see Bertsekas and Tsitsiklis 1996 and Sutton and Barto 1998 for an introduction to the techniques of approximate dynamic programming). The problem is that in realistic problems, the attribute space $\mathcal{A}$ can be extremely large, and we may obtain only a few observations of $\hat{v}_{a'}$ for a particular a'. As a result, the statistical error in $\bar{v}_{a'}$ can be quite large.

One of the standard strategies in approximate dynamic programming is to aggregate the state (attribute) space. Instead of estimating $\bar{v}_a$, we might define an aggregation function $G(a)$ which produces an aggregated attribute $\bar{a}$ which has fewer outcomes. For example, a five-digit zip code can be aggregated up to a three-digit zip; a numerical attribute can be divided into fewer ranges; or an attribute can be completely ignored. The resulting smaller attribute space produces more observations of each attribute, but at a cost of aggregation error.

There are a variety of statistical strategies for estimating value functions which take advantage of the structure of a specific attribute vector a. In a trucking problem, we might design a statistical function that depends on the location of a driver, his days away from home, the fuel level of his tank and his home domicile. However, after designing a statistical model that works for this application,

we would have to start from scratch if we wished to switch to another application. In fact, simply adding an attribute would require redesigning and refitting the statistical equation. This can be particularly hard when several of the attributes are categorical, and which interact to determine the effect of the attributes on the system. A truck driver might be characterized by his location and his home domicile; the value of a driver at a location depends very much on where he lives.

We are interested in developing a method for estimating the value $\bar{v}_a$ of a resource with attribute a, making minimal assumptions about the structure of the attribute space. We take advantage of the fact that for every application with which we are familiar, it is quite easy to design a family of aggregation functions $\mathcal{G}$ where $G^{(g)} : \mathcal{A} \rightarrow \mathcal{A}^{(g)}$ is an aggregation of the attribute space $\mathcal{A}$. For example, we can create an aggregation function simply by ignoring an attribute. Aside from assuming the existence of this family of functions, we make no further assumptions about the nature of the attribute space. For example, we do not even require the existence of a metric that would provide a measure of the distance between two attribute vectors, which prevents the use of standard methods such as non-parametric statistics or regression trees.

Aggregation has traditionally been a powerful technique in dynamic programming. A good general review of aggregation techniques is given by Rogers et al. (1991). Aggregation strategies in a dynamic programming setting may be governed by the desire to solve exactly a smaller dynamic program, or by the iterative nature of the algorithms. Techniques range from picking a fixed level of aggregation (Whitt, 1978; Bean et al., 1987; Athans et al., 1995; Zhang and Sethi, 1998; Wang and Dietterich, 2000), or using adaptive techniques that change the level of aggregation as the sampling process progresses (Mendelssohn, 1982; Bertsekas and Tsitsiklis, 1996; Luus, 2000; Kim and Dean, 2003), but which still use a single level of aggregation at any given time (many authors used a fixed level of aggregation to produce a smaller Markov Decision Process (MDP) that can be solved optimally). Tsitsiklis and Van Roy (1996) (see also Bertsekas and Tsitsiklis, 1996) show how value functions can be approximated using a fixed set of *features*; this strategy encompasses both static and hierarchical aggregation as special cases, but the use of these techniques in our setting is prohibitive because of the extremely large number of values that need to be estimated. Feng et al. (2003) presents a work that identifies state aggregations based on "structural similarity" where states are considered similar if they have similar value estimates or similar sets of successor states, rather than "input similarity" which is typically measured by some distance metric defined over the state space. Bertsekas and Castanon (1989) introduces a creative approach which adaptively clusters states with similar values of residual errors at each iteration, requiring no structure among the states of the system. While we also do not have any structure, we do take advantage of the presence of a family of aggregation functions, and our technique does not require the overhead of solving clustering problems. A nice discussion of aggregation and abstraction techniques in an approximate dynamic programming setting is given in Boutilier et al. (1999).

Instead of using a single level of aggregation, researchers have considered combining estimates from all the levels of aggregation at once. In the literature, there exist several techniques for combining estimates to improve accuracy (see Wolpert, 1992; LeBlanc and Tibshirani, 1996; Yang, 2001). It is well-known that if the estimates being combined are independent and unbiased, then it is optimal to combine them in inverse proportion to their variances (Guttman et al., 1965). When different estimates are based on different levels of aggregation, they are neither independent nor unbiased. It is also possible to use a weighted combination of estimates where the weights are estimated using regression techniques. For our application, there can be hundreds of thousands of such models, making the updating of regression models computationally expensive.

In this paper, we solve the problem of optimally combining (correlated) value estimates at different levels of aggregation in an approximate dynamic programming setting and derive expressions for optimal weights. The result generalizes a well-known result for optimally combining independent estimates. We point out that the independence assumptions used for deriving the results are true only in idealized regression settings, and not in an approximate dynamic programming setting.

The major contribution of this paper lies in finding that an inverse-variance weighting formula (adjusted for bias), which is optimal only when the estimates are independent, proves to be near-optimal even though estimates at different levels of aggregation are not independent. We explain this behavior analytically for the case with two levels of aggregation. We show that if we compute optimal weights (without assuming independence) and compare the results if we do assume independence, the results are the same for two extremes: when the difference between the aggregate and disaggregate value estimates is very large or very small. We show experimentally that the error for intermediate values is extremely small.

We also show, in the context of a single vehicle routing problem, that our weighting method produces value function estimates that are within five to ten percent of the optimal value functions, outperforming other estimates. The method of weighting a family of aggregate estimates is shown to naturally shift the weight from aggregate to disaggregate estimates as the algorithm progresses. We also demonstrate that this method is easy to implement in large-scale, on-line learning applications that arise in approximate dynamic programming, where it produces much faster convergence (which implies approaching a consistently better solution quality in a fewer number of iterations) than would be produced using a single, static level of aggregation. Further work on this application is explained in detail in Simao et al. (2008).

The paper is organized as follows. In Section 2, we describe a generic approximate dynamic programming technique, which estimates the value functions associated with various states. This section provides an introduction to the context in which our statistical estimation problem arises. The next three sections, however, focus purely on the statistics of aggregation outside of a dynamic programming setting. Section 3 provides a theoretical model of the sampling process and defines bias and variance for aggregated statistics. Then, Section 4 poses the problem of computing optimal weights for combining estimates of values at different levels of aggregation. The problem with this formula is that it is too expensive to use for our problem class. For this reason, we propose a simpler formula that assumes that statistics from different levels of aggregation are independent. In Section 5, we compare the two weighting formulas (with and without the independence assumption) for the special case where there are only two levels of aggregation which allows the optimal weights to be computed analytically. We show theoretically that assuming independent estimates introduces zero expected error at two extremes of the problem. We then show experimentally that ignoring the dependence between the estimates gives results that are very similar. In Section 6, we demonstrate our approximation method in the context of an approximate dynamic programming algorithm for solving a multiattribute resource allocation problem. We use both a single truck problem, which can be solved exactly, as well as a problem of managing a large fleet of trucks. We provide our concluding remarks in Section 7.

2. Approximate Dynamic Programming

This section is designed as a brief introduction to approximate dynamic programming, and introduces the context in which our problem arises. Our interest lies in the context of dynamic resource

allocation problems. Dynamic programming techniques can be applied to solve these problems which are typically modeled as MDPs. Using the notation of Powell (2007), we let S_t be the state of our system. We also let $d \in \mathcal{D}$ be a type of a decision, and we let $x_d = 1$ if we choose decision d, and $x_d = 0$ otherwise. $x_t = (x_d)_{d \in \mathcal{D}}$ is the vector of decisions that we make at time t. Bellman's equation allows us to express the value of being in state S_t as

$$V(S_t) \quad = \quad \max_{x_t} \left[C(S_t, x_t) + \mathbb{E} \left\{ V \left(S_{t+1}(S_t, x_t, W_{t+1}) \right) | S_t \right\} \right],$$

where W_{t+1} is a random variable representing new information that arrives between t and $t+1$. The exact values can be determined using traditional backward dynamic programming techniques such as value iteration and policy iteration. In these methods, the values are computed recursively starting from the final state, making use of the state transition probabilities.

When the state and action spaces become large, as in most real-life stochastic planning problems, it is not practical to enumerate the states to determine their values. In such problems, compact feature-based representations of the MDP, also called factored MDPs (see Boutilier et al., 2000) can be used to make the problem computationally tractable. Factored MDPs can be represented using a factored state transition model and a reward function that is additive. In these representations, a smaller set of variables (also called features or attributes) are used to describe the state of the system.

Dynamic resource allocation problems span dynamic vehicle routing (Gendreau and Potvin, 1998; Ichoua et al., 2005), where there has been recent interest in the application of approximate dynamic programming for the single vehicle routing problem (Secomandi, 2000, 2001). Powell and Carvalho (1998) uses an approximate dynamic programming algorithm for a fleet management problem, but the attributes of the vehicles were very simple. Powell et al. (2002) uses an approximate dynamic programming algorithm for multiattribute resources, but does not address statistical sampling issues. Spivey and Powell (2004) applies approximate dynamic programming for optimizing a fleet of vehicles, using a linear value function approximation that also requires estimating the value of a resource characterized by a vector of attributes. This research estimated the value of a resource at different levels of aggregation, but kept track of the variance of these estimates at each level of aggregation and always used the estimate that provided the smallest variance.

Resource allocation problems can be modeled by letting $a \in \mathcal{A}$ be an attribute vector (a may consist of categorical and numerical attributes), and by letting R_{ta} be the number of resources with attribute a. We then let $R_t = R_{ta})_{a \in \mathcal{A}}$ be the resource state vector. This research addresses problems where the vector a is large enough that the attribute space $\mathcal{A}$ becomes too large to enumerate. We develop these ideas in the context of a single entity. If a_t is the attribute of the entity at time t, then a_t is effectively our state variable.

In this section, we describe the basic approximate dynamic programming (ADP) strategy to solve the problem of managing a single resource with multiple attributes, the *nomadic trucker*. This is a single resource version of the dynamic fleet management problem, where there is a single trucker who needs to move between various locations to cover loads that arise and gains rewards in the process.

The state of the resource is defined by an attribute vector, a, composed of multiple attributes, which may be numerical or categorical. For the nomadic trucker problem, examples of attributes include the location of the truck, the home domicile of the driver and the number of hours driven. We could represent the attribute vector as $a = (a_{location}, a_{time}, a_{domicile}, \ldots)$. The state space, $\mathcal{A}$, for

this problem would consist of all possible combinations of the attributes of the trucker. We can let the decision be represented by the vector $(x_d)_{d\in\mathcal{D}}$, but for a single entity problem, $\sum_{d\in\mathcal{D}} x_d = 1$, which means we can also write the problem as choosing a decision $d \in \mathcal{D}$. Typically, the set of potential decisions depends on the current state (attributes) of our resource, so we let $\mathcal{D}_a$ be the decisions available to a resource with attribute a. We assume that the impact of a decision d on a resource with attribute a is deterministic, and is given by the function $a' = a^M(a,d)$.

In approximate dynamic programming, we sample the various states by choosing decisions that are locally optimal based on current estimates of the value functions. For example, we could follow a procedure where we choose a decision that maximizes the sum of the one-period rewards and the future value (discounted by factor γ) as follows:

$$d(a,\omega) \quad = \quad \arg\max_{d\in\mathcal{D}_a(\omega)} \left\{ c(a,d,\omega) + \gamma v_{a^M(a,d)} \right\}.$$

Here, ω represents a sample realization of random information (for example, $\mathcal{D}_a(\omega)$ is a sample realization of the decision set), $a^M(a,d)$ is the state at the destination and $v_{a^M(a,d)}$ the value associated with $a^M(a,d)$. This model is easily generalized to handle stochastic transitions, but this is not relevant to the focus of this paper.

We outline the steps of a typical approximate dynamic programming algorithm for the nomadic trucker problem in Figure 1. This algorithm has two stages. In the forward pass, we use the current estimates of the optimal value functions to simulate a sample trajectory of the truck. The next state that is visited is determined using a transition function $a^M(a_m,d_m)$, as depicted in Equation 2, where the resource in state a_m undergoes a transformation to state $a_{m+1} = a^M(a_m,d_m)$ when acted upon by decision d_m. Once the end of the time horizon is reached, we perform a backward pass, where we first compute the observations of values of the various states in the current sample path using Equation 3. We point out that the estimates of the future values are discounted by a factor γ. We then use these to update the value estimates, as in Equation 4, and the associated statistics (number of observations and sample variance) of the states that are visited.

There are a number of variations of approximate dynamic programming. One family is known as $TD(\lambda)$-learning (see Sutton, 1988; Sutton and Barto, 1998), typically parameterized by an artificial discount factor λ. Using a pure forward pass algorithm is equivalent to $TD(0)$, while another variation follows a policy (determined by the current set of approximations), and then does a backward traversal to obtain updates of the estimate of the value of being in each state (this is equivalent to $TD(1)$). Another popular strategy is Q-learning (see Watkins, 1989), where we estimate the quantities $Q(a,d)$ which is the value of being in a state a and making decision d. Since the statistical problem of estimating the value of a state-action pair is, of course, even harder than the problem of estimating the value of being in a state, we have not used this approach. Since Q-learning allows you to determine a decision directly from the Q-factors (rather than solving an optimization problem), it is typically presented as a "model-free" algorithm (that is, one that does not require an explicit model of the transition function), although estimating the Q-factors does require some source that determines the next state given a state and action. All of these methods can be used without an explicit model of the exogenous information process (for example, we do not use a one-step transition function) as long as we have some mechanism for creating the sample realizations.

As with most ADP algorithms, the only way to obtain an estimate of the value of being in a state is to actually visit the state. In real applications, there may be millions of states but we may be limited to only thousands of observations. In practice, most states are never visited, and many are

Step 0. Initialize an approximation for the value function $\bar{v}_a^0$ for all attribute vector states $a \in \mathcal{A}$ and set $n = 1$.

Step 1. Iteration n:

Step 2. *Forward pass*: Set $m = 0$ and randomly sample attribute vector a_m, but fixing the start time at the beginning of the time horizon.

Step 3. Obtain the set of possible decisions, $\mathcal{D}_{a_m}(\omega)$.

Step 4. Solve for the optimal decision, given the current value function estimates.

$$d_m(\omega) = \arg\max_{d \in \mathcal{D}_{a_m}(\omega)} \left[c(a_m, d) + \gamma \bar{v}^{n-1}_{a^M(a_m,d)} \right] \quad (1)$$

Step 5. Evaluate the next state to visit:

$$a_{m+1} = a^M(a_m, d_m) \quad (2)$$

Step 6. If the end of the time horizon (T) is reached, then set $m = m+1$ and go to **step 3**, else go to **step 7**.

Step 7. *Backward pass*: For $j = m-1, m-2, \cdots, 0$, update the value function estimates as follows:

$$\hat{v}^n_{a_j} = c(a_j, d_j) + \gamma \hat{v}^n_{a_{j+1}} \quad (3)$$

$$\bar{v}^n_{a_j} = (1-\alpha)\bar{v}^{n-1}_{a_j} + \alpha \hat{v}^n_{a_j} \quad (4)$$

Step 8. Let $n = n+1$. If $n < N$ go to **step 1**, else for each state a, return the value function $\bar{v}^n_a$.

Figure 1: An approximate dynamic programming algorithm using a backward pass for the nomadic trucker problem

visited only a few times. As a result, there can be a high level of statistical noise in our estimates of the value of being in a state.

This section provides the context in which our adaptive learning problem arises. The next three sections consider the general problem of estimating a quantity (the value of a resource with attribute a) outside of the context of approximate dynamic programming. We assume we have a source of (unbiased) observations of the value associated with attribute a, from which we have to develop statistically robust (i.e., low-variance) estimates of the value associated with attribute a. We then use the method in the context of approximate dynamic programming to demonstrate that it produces better results than other methods, even though we no longer have unbiased observations.

3. The Statistics of Aggregation

In this section, we investigate the statistics of aggregation by studying a sampling process where at iteration n we first sample the attribute vector $a = \hat{a}^n$. We then use a sample realization of the random information which provides us with an unbiased observation of the value of the resource $\hat{v}^n$, producing a sequence of observations of (attribute vector, value) pairs. We wish to use this information to produce a statistically reliable estimate of the true value associated with a. The analysis in this section is not done in the context of dynamic programming (which allows us to assume that our observations of values are unbiased). Rather, it is intended as a pure study of the statistics of aggregation.

Our assumption that the observations of values, $\hat{v}^n$, are unbiased will not be true in a dynamic programming setting, but allows us to focus on the tradeoff between bias and variance.

We begin by defining the following:

$\mathcal{N} =$ The set of indices corresponding to the observations of the attribute vectors and values.

$\mathcal{S} =$ A sample of observations $(\hat{a}^n, \hat{v}^n)_{n \in \mathcal{N}}$.

$\nu_a =$ The true value associated with attribute vector a.

$N_a =$ The number of observations of attribute vector a given our sample $\mathcal{S}$.

$\hat{a}^n =$ The attribute vector at observation n.

$\hat{v}^n =$ The observation of the value corresponding to index n.

$1_{\{\hat{a}^n = a\}} =$ 1, if the nth observation is of attribute vector a.

An estimate of ν_a can be obtained as an average across all the observations of values corresponding to a:

$$\bar{v}_a = \frac{1}{N_a} \sum_{n \in \mathcal{N}} \hat{v}^n 1_{\{\hat{a}^n = a\}}.$$

Throughout our presentation, we use the hat notation (as in $\hat{v}$) to represent exogenous information, and bars (as in $\bar{v}$) to represent statistics derived from exogenous information.

Consider a case where the attribute vector has more than one dimension, with A_i denoting the number of distinct states that attribute a_i can assume. The number of values that need to be estimated is $\prod_i A_i$. Needless to say, as the attribute vector grows, the state space grows exponentially, making it impossible to obtain statistically reliable estimates. One strategy is to resort to aggregation (such as dropping one or more dimensions of a) which can quickly reduce the number of values but introduces structural error. An alternative is to assume a structural property such as separability, which reduces the number of values to be estimated to $\sum_i A_i$. This has fewer values, but requires that we introduce separability as an approximation. In one of our trucking applications, one attribute is the location of the truck, while a second attribute is the driver's home domicile. The value of a driver in a location depends very much on his home domicile. Assuming these are independent would introduce significant errors.

In general, aggregation of attribute vectors is performed using a collection of aggregation functions, $G^g : \mathcal{A} \rightarrow \mathcal{A}^{(g)}$, where $\mathcal{A}^{(g)}$ represents the g^{th} level of aggregation of the attribute space $\mathcal{A}$. We define the following:

$a^{(g)} =$ $G^g(a)$, the g^{th} level aggregation of the attribute vector a.

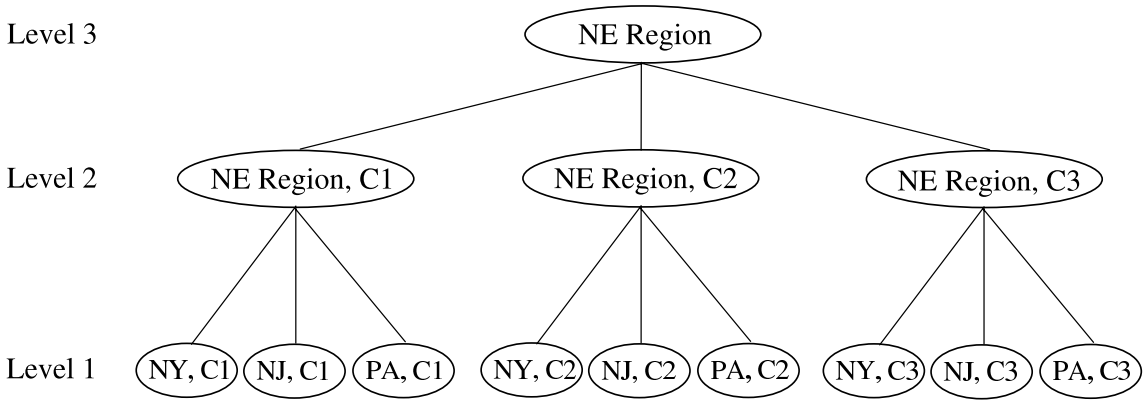


Figure 2: Aggregation of the state space for a multiattribute problem.

$\mathcal{G} =$ The set of indices corresponding to the levels of aggregation.

Aggregation can thus be used to create a sequence of state spaces, $\left\{\mathcal{A}^{(g)}, g = 1,2,\ldots,|\mathcal{G}|\right\}$, with fewer elements than the original state space. This can be better illustrated using the example in Figure 2, where we consider the nomadic trucker problem with the state of the truck defined by two attributes - current location and capacity type. The number of possible states with three locations (NY, NJ and PA), and three capacity types (C1, C2 and C3), is nine at the most disaggregate level. The first-level aggregation function, $G^{(1)}$, involves aggregating the location to the regional level which reduces the number of states to three. The second-level aggregation function, $G^{(2)}$, would be defined as aggregating out the capacity type attribute completely, which leaves us with a single state. As in this example and in the experimental work to follow, it is usually the case that the gth level of aggregation acts on the $(g-1)$st level.

We let ε^n denote the error in the nth observation with respect to the true value associated with $\hat{a}^n$ (which, using the notation defined earlier in this section, would be represented using $\nu_{\hat{a}^n}$). For analysis purposes, we assume that the elements of the sequence $\{\varepsilon^n\}_{n\in\mathcal{N}}$ are independent and identically distributed, with a mean value of zero. This is, of course, an idealization, but it will help us understand the tradeoffs between structural errors (due to aggregation) and statistical errors. We can express the observed value as follows:

$$\hat{v}^n = \nu_{\hat{a}^n} + \varepsilon^n.$$

We define the following probability spaces,

$\Omega^a =$ The set of outcomes of observations of attribute vectors.

$\Omega^\varepsilon =$ The set of outcomes of observations of the errors in the values.

$\Omega =$ The overall set of outcomes

$= \Omega^a \times \Omega^\varepsilon.$

$\omega =$ $(\omega^a, \omega^\varepsilon)$

$=$ An element of the outcome space.

We now define the following terms which will be useful in obtaining an estimate of the value associated with the attribute vector a at any level of aggregation:

$\mathcal{N}_a^{(g)} =$ The set of indices that correspond to observations of the attribute vector a at the gth level of aggregation

$= \{n \mid G^g(\hat{a}^n) = G^g(a)\}.$

$N_a^{(g)} = \left|\mathcal{N}_a^{(g)}\right|.$

$\bar{v}_a^{(g)} =$ The estimate of the value associated with the attribute vector a at the gth level of aggregation, given the sample, $\mathcal{N}$.

We can compute the estimate, $\bar{v}_a^{(g)}$, as

$$\bar{v}_a^{(g)} = \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \hat{v}^n.$$

We provide a numerical example to illustrate the idea of forming estimates at different levels of aggregation. Consider the state of a resource to be composed of two attributes, namely, location of the resource and resource type. There are four locations, namely, New York, Philadelphia, Boston and Washington. The type can be Single or Team. Thus, there are eight possible states. We use aggregation functions that aggregate out the type attribute and then the location attribute to obtain three different levels of aggregation. Suppose we have the following observations of state-value

a	Location	Type	N_a	$\bar{v}_a$	$N_a^{(1)}$	$\bar{v}_a^{(1)}$	$N_a^{(2)}$	$\bar{v}_a^{(2)}$
a_1	New York	Single	2	4.5				
a_2	New York	Team	1	7.0	3	5.3		
a_3	Philadelphia	Single	3	3.7				
a_4	Philadelphia	Team	1	2.0	4	3.3	12	4.8
a_5	Boston	Single	2	8.5				
a_6	Boston	Team	0	-	2	8.5		
a_7	Washington	Single	1	1.0				
a_8	Washington	Team	2	5.5	3	4.0		

Table 1: Numerical example illustrating the computation of value estimates using aggregation. For example, $\bar{v}_{a_1}^{(0)} = (7+2)/2 = 4.5$, and $\bar{v}_{a_7}^{(1)} = (5+1+6)/3 = 4.0$.

pairs - $\{(a_3,4),(a_4,2),(a_1,7),(a_5,8),(a_3,2),(a_8,5),(a_7,1),(a_5,9),(a_8,6),(a_2,7),(a_3,5),(a_1,2)\}$. We can form estimates of the values of the various states at the different levels of aggregation as illustrated in Table 1.

Now that we have an estimate of the value, v_a, for each level of aggregation, the question arises as to what is the best level of aggregation. A traditional strategy is to choose the right level of aggregation by trading off statistical and structural errors to find a model with the least overall error.

In order to better understand these two kinds of errors in an aggregation setting, we first let $\overline{\delta}_a^{(g)}$ denote the total error in the estimate, $\overline{v}_a^{(g)}$, from the true value associated with attribute vector a:

$$\overline{\delta}_a^{(g)} = \overline{v}_a^{(g)} - \nu_a.$$

An important component of our prediction error will be aggregation bias. Consider our most recently observed attribute vector $\hat{a}^n$ and some other attribute a, where $\hat{a}^n$ and a may aggregate up to the same aggregated attribute at some level $g \in \mathcal{G}$, that is, $G^g(a) = G^g(\hat{a}^n)$ (for the moment, these are simply two attribute vectors). In our derivations below, it is useful to define a bias term,

$$\mu_a^n = \nu_{\hat{a}^n} - \nu_a.$$

We can use this notation to rewrite $\hat{v}^n$ as follows:

$$\begin{aligned} \hat{v}^n &= \nu_a + (\nu_{\hat{a}^n} - \nu_a) + \varepsilon^n \\ &= \nu_a + \mu_a^n + \varepsilon^n \quad \forall\, a, n. \end{aligned}$$

We can express $\overline{v}_a^{(g)}$ in terms of its bias and noise components as follows:

$$\begin{aligned} \overline{v}_a^{(g)} &= \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} (\nu_a + \mu_a^n + \varepsilon^n) \\ &= \nu_a + \left(\frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \mu_a^n \right) + \left(\frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \varepsilon^n \right). \end{aligned}$$

We let,

$$\begin{aligned} \overline{\mu}_a^{(g)} &= \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} \mu_a^n, \\ \overline{\varepsilon}_a^{(g)} &= \frac{1}{N_a^{(g)}} \sum_{n \in \mathcal{N}_a^{(g)}} .\varepsilon^n. \end{aligned}$$

This enables us to express the total error as follows:

$$\overline{\delta}_a^{(g)} = \overline{\mu}_a^{(g)} + \overline{\varepsilon}_a^{(g)} \tag{5}$$

where $\overline{\mu}_a^{(g)}$ gives an estimate of the bias between the values of a at the gth level of aggregation and at the disaggregate level. $\overline{\mu}_a^{(g)}$ is a random variable that is a function of the set of points sampled. $\overline{\varepsilon}_a^{(g)}$ is an estimate of the random error that has zero expected value. By assumption, the variability in $\overline{\varepsilon}_a^{(g)}$ occurs because of the statistical noise in the observation of the values. We point out that the terms, $\overline{\delta}_a^{(g)}$, $\overline{\mu}_a^{(g)}$ and $\overline{\varepsilon}_a^{(g)}$, are not statistical estimators, because a knowledge of the true values is required for computing these. $\overline{\mu}_a^{(g)}$ is representative of the structural error that is introduced due to aggregation, while $\overline{\varepsilon}_a^{(g)}$ represents the statistical error due to noise in the observations. Moreover, these two error terms need not be uncorrelated in a general setting.

In an approximate dynamic programming setting, the right tradeoff between statistical and structural errors will change as we collect more observations. Furthermore, we generally do not control the sampling process of the attributes, and we will encounter instances where some regions of the attribute space $\mathcal{A}$ will be sampled more than others. Although it is common in practice to choose a single level of aggregation that produces the lower overall error, it can be useful to combine estimates from several levels of aggregation.

4. Combining Estimates

In this section, we propose methods to compute weights to combine value estimates that have been formed from a given set of observations. In the context of ADP, the weights are computed at a given iteration of the algorithm in Figure 1.

We consider a set of estimates, $\left\{\bar{v}_a^{(g)}, g \in \mathcal{G}\right\}$, of a value, ν_a, at different levels of aggregation. We let $\sigma_a^{(g)}$ denote the population standard deviation associated with the observations used to compute $\bar{v}_a^{(g)}$. Breiman (1996) proposes a method called stacked regression which in our setting would be equivalent to combining estimates at different levels of aggregation using

$$\bar{v}_a = \sum_{g \in \mathcal{G}} w^{(g)} \cdot \bar{v}_a^{(g)},$$

where $w^{(g)}$ is a set of weights for each level of aggregation. This method ignores the important feature that the best weighting depends on how many times we have observed a particular attribute. We prefer to use the strategy suggested by LeBlanc and Tibshirani (1996) (Section 8), where the weights depend on the attribute:

$$\bar{v}_a = \sum_{g \in \mathcal{G}} w_a^{(g)} \cdot \bar{v}_a^{(g)}.$$

The practical challenge here is that we have to estimate a set of weights $(w_a^{(g)})$ for *each* attribute a (that we observe). If we use classical regression methods for our applications, this can mean maintaining hundreds of thousands of regression models. Storing and updating these models is computationally demanding for large industrial applications. In this section, we develop both exact and approximate methods for estimating weights, where our approximation makes the assumption that the estimates $\bar{v}_a^{(g)}$ are independent. Section 5 presents theoretical and experimental arguments supporting the accuracy of the weights when we assume independence (even when the assumption is not even approximately true), which dramatically simplifies the procedure.

In Section 4.1, we formulate the problem of finding the optimal weights for the general case where the estimates may be biased and dependent on each other, and later derive these weights. However, the computation of these weights can prove cumbersome for large-scale problems. We provide, in Section 4.2, a simpler formula for the weights which assumes that the estimates are independent, but accounts for the possibility of biases in the estimates. In Section 4.3, we propose an approximation of the weights derived in Section 4.2, for the case where the bias and variance of the estimators are unknown.

4.1 Optimal Weights

We begin by finding the weighting scheme that will optimally combine the estimates at the different levels of aggregation, that is, the weights which give a combined estimate with the least squared deviation from the true value associated with attribute vector a. We can formulate the problem as follows:

$$\min_{w_a^{(g)}, g \in \mathcal{G}} \mathbb{E}\left[\frac{1}{2}\left(\sum_{g \in \mathcal{G}} w_a^{(g)} \cdot \bar{v}_a^{(g)} - \nu_a\right)^2\right], \tag{6}$$

subject to:

$$\sum_{g \in \mathcal{G}} w_a^{(g)} = 1. \tag{7}$$

In a setting where the estimates are unbiased, it is useful to have an affine combination of the estimates (LeBlanc and Tibshirani, 1996, Section 2) since the individual estimates and hence the affine combination are equal to the true value in expectation. Even though this is not necessarily true in a general setting, we choose to retain this constraint.

We state the following proposition for computing the optimal weights that solves the problem formulated in Equations 6-7:

Proposition 1 *For a given attribute vector, a, the optimal weights, $w_a^{(g)}$, $g \in \mathcal{G}$, to combine individual estimates that are correlated in a hierarchical fashion, are obtained by solving the following system of linear equations in (w, λ):*

$$\sum_{g \in \mathcal{G}} w_a^{(g)} \mathbb{E}\left[\bar{\delta}_a^{(g)} \bar{\delta}_a^{(g')}\right] - \lambda = 0 \quad \forall\ g' \in \mathcal{G}, \tag{8}$$

$$\sum_{g \in \mathcal{G}} w_a^{(g)} = 1. \tag{9}$$

If the bias error, $\bar{\mu}_a^{(g)}$, is uncorrelated with the random error, $\bar{\varepsilon}_a^{(g)}$, then the coefficients of the weights in Equation 8 can be expressed as follows:

$$\mathbb{E}\left[\bar{\delta}_a^{(g)} \bar{\delta}_a^{(g')}\right] = \mathbb{E}\left[\bar{\mu}_a^{(g)} \bar{\mu}_a^{(g')}\right] + \frac{\sigma_\varepsilon^2}{N_a^{(g')}} \qquad \forall g \leq g' \text{ and } g, g' \in \mathcal{G} \tag{10}$$

where σ_ε^2 denotes the variance of the statistical noise in the observations.

Proof: The proof is given in appendix A. The derivation of Equation 8 involves using the Lagrangian for the problem stated in Equations 6-7 and performing some simple arithmetic on the corresponding first order optimality conditions. Equation 9 is identical to Equation 7 from the optimization formulation.

In the remainder of this analysis, our computations will be conditional on a given sequence of observed attribute vectors. In other words, all expectations and probabilities are computed with respect to the probability space, Ω^ε. We prove Equation 10 by simplifying the expression $\mathbb{E}\left[\bar{\delta}_a^{(g)} \bar{\delta}_a^{(g')}\right]$ using some properties of hierarchical aggregation. □

For the case where $g = 0$, we can use the result, $\mathbb{E}\left[\bar{\mu}_a^{(0)}\bar{\mu}_a^{(g')}\right] = 0$ (which follows from the property: $\mu_a^{(0)} = 0$), to further simplify (10) and obtain the following result:

$$\mathbb{E}\left[\bar{\delta}_a^{(0)}\bar{\delta}_a^{(g')}\right] = \frac{\sigma_\varepsilon^2}{N_a^{(g')}}. \tag{11}$$

We refer to the optimal weighting scheme as WOPT.

4.2 An Approximation Assuming Independence

It is a well-known result in statistics that if the estimates $\left\{\bar{v}_a^{(g)}, g \in \mathcal{G}\right\}$ were independent and unbiased, then the optimal weights would be given by

$$w_a^{(g)} = \frac{1}{{\sigma_a^{(g)}}^2/N_a^{(g)}}\left(\sum_{g' \in \mathcal{G}} \frac{1}{{\sigma_a^{(g')}}^2/N_a^{(g')}}\right)^{-1}. \tag{12}$$

We can obtain this result from proposition 1 as follows. If we assume that the estimates $\left\{\bar{v}_a^{(g)}, g \in \mathcal{G}\right\}$ are independent and unbiased, then the cross-terms in Equation 8 disappear, leaving behind the following modified relation:

$$w_a^{(g)}\mathbb{E}\left[\left(\bar{\delta}_a^{(g)}\right)^2\right] - \lambda = 0 \quad \forall\ g \in \mathcal{G}. \tag{13}$$

Solving Equations 13 and 9 gives us weights that are inversely proportional to the expected squared errors, $\mathbb{E}\left[\left(\bar{\delta}_a^{(g)}\right)^2\right]$. For the case of independent, unbiased estimates, $\mathbb{E}\left[\left(\bar{\delta}_a^{(g)}\right)^2\right]$ is identical to the variance, ${\sigma_a^{(g)}}^2/N_a^{(g)}$.

Solving the system of equations in Proposition 1 can be computationally expensive since in practice, there may be hundreds of thousands of models. For practical solutions, it will be useful to have an expression along the lines of Equation 12 for computing the weights, even though neither of the conditions (independence and absence of bias) holds true for estimates that arise from aggregation due to structural errors introduced in the process of aggregation. In order to adapt the simpler formula in (13) to the aggregation setting while acknowledging the bias, we first define:

$$\begin{aligned}\mu_a^{(g)} &= \text{Expected bias in the estimate, } \bar{v}_a^{(g)} \\ &= \mathbb{E}\left[\bar{v}_a^{(g)} - \nu_a\right].\end{aligned}$$

For biased estimates, the total squared error can be expressed as the sum of bias and variance components, provided the bias and variance are independent of each other (Hastie et al., 2001, p. 24):

$$\mathbb{E}\left[\left(\bar{\delta}_a^{(g)}\right)^2\right] = \frac{{\sigma_a^{(g)}}^2}{N_a^{(g)}} + {\mu_a^{(g)}}^2. \tag{14}$$

We use this relation to modify the weights as follows:

$$w_a^{(g)} = \frac{1}{\frac{{\sigma_a^{(g)}}^2}{N_a^{(g)}} + {\mu_a^{(g)}}^2} \left(\sum_{g' \in \mathcal{G}} \frac{1}{\frac{{\sigma_a^{(g')}}^2}{N_a^{(g')}} + {\mu_a^{(g')}}^2} \right)^{-1} \quad \forall\ g \in \mathcal{G}. \tag{15}$$

We call this weighting scheme, WIND.

4.3 Weighting by Inverse Mean Squared Errors

In the more realistic setting where the exact values of the parameters involved in the computation of weights as in Equation 15 are unknown, we propose using the plug-in principle (see, for example, Efron and Tibshirani 1993, chapter 4) where we use statistical estimates of the bias and variance to produce approximations of the weights. We first compute estimates of the bias and the variance using

$$\begin{aligned} {s_a^{(g)}}^2 &= \text{The sample variance of the observations corresponding to the estimate } \bar{v}_a^{(g)} \\ &= \frac{1}{N_a^{(g)} - 1} \sum_{n \in \mathcal{N}_a^{(g)}} \left(\hat{v}^n - \bar{v}_a^{(g)} \right)^2. \\ \tilde{\mu}_a^{(g)} &= \text{An estimate of the bias in the estimated value } (\bar{v}_a^{(g)}) \text{ from the true value} \\ &= \bar{v}_a^{(g)} - \bar{v}_a^{(0)}. \end{aligned}$$

The approximate weights on the estimates at different levels of aggregation are inversely proportional to the estimates of their mean squared deviations (obtained as the sum of the variances and the biases) from the true value:

$$w_a^{(g)} = \frac{1}{\frac{{s_a^{(g)}}^2}{N_a^{(g)}} + {\tilde{\mu}_a^{(g)}}^2} \left(\sum_{g' \in \mathcal{G}} \frac{1}{\frac{{s_a^{(g')}}^2}{N_a^{(g')}} + {\tilde{\mu}_a^{(g')}}^2} \right)^{-1} \quad \forall\ g \in \mathcal{G}. \tag{16}$$

We refer to this formula as weighting by inverse mean squared errors (WIMSE). In the event that $N_a^{(g)}$ is too small or zero (which can happen in the early iterations and/or at the more disaggregate levels), it is difficult to form meaningful estimates of the variance and bias. In such a situation, we set the corresponding weight to zero.

Equation 16 is very easy to calculate even for large scale applications where we may observe hundreds of thousands of attributes. However, it produces the best results only when the estimates of values at different levels of aggregation are independent, an assumption that we cannot expect to hold true. In the next section, we present theoretical and experimental evidence supporting the claim that the error introduced from this assumption is negligible.

It is important to note that the use of the plug-in principle, which in this setting means using statistical estimates of parameters (the bias and variance), may result in some unexpected behavior when the number of observations is small. For example, the estimate of the total squared error in Equation 14 would be expected to decrease with each additional observation. When we use estimates of the bias and variance, this is no longer guaranteed, especially when $N_a^{(g)}$ is small. However, our empirical evidence is that it seems to behave as expected in an aggregate sense.

5. The Case for Assuming Independence

In this section, we justify our decision to ignore the dependence between the estimates from hierarchical aggregation, while combining them to form an improved estimate. We discuss the special case where we combine estimates from only two levels of aggregation, which enables us to obtain simple expressions for computing the various parameters. We assume that the statistical noise is independent of the attribute vector sampled and also that we know the probability distributions of the sampling of the attribute vectors and their values. These assumptions enable us to solve the optimality equations to obtain a solution explicitly. In Section 5.1, we analytically compare the two sets of equations (with and without assuming independence) for computing optimal weights. We provide an experimental comparison of the two methods, demonstrating the similarity in results, in section 5.2.

5.1 Analytical Comparison

For the two-level problem, we can obtain the optimal weights (WOPT) by solving the following system of equations:

$$\begin{aligned}
\mathbb{E}\left[\overline{\delta}_a^{(0)^2}\right] w_a^{(0)} + \mathbb{E}\left[\overline{\delta}_a^{(0)}\overline{\delta}_a^{(1)}\right] w_a^{(1)} - \lambda &= 0, \\
\mathbb{E}\left[\overline{\delta}_a^{(0)}\overline{\delta}_a^{(1)}\right] w_a^{(0)} + \mathbb{E}\left[\overline{\delta}_a^{(1)^2}\right] w_a^{(1)} - \lambda &= 0, \\
w_a^{(0)} + w_a^{(1)} &= 1, \\
w_a^{(0)}, w_a^{(1)} &\geq 0.
\end{aligned}$$

Since we are concerned with computing the weights for a particular attribute vector, we drop the index a in the following analysis. We obtain the value of $w^{(0)}$ as,

$$w^{(0)} = \frac{\mathbb{E}\left[\overline{\delta}^{(1)^2}\right] - \mathbb{E}\left[\overline{\delta}^{(0)}\overline{\delta}^{(1)}\right]}{\mathbb{E}\left[\overline{\delta}^{(0)^2}\right] + \mathbb{E}\left[\overline{\delta}^{(1)^2}\right] - 2\mathbb{E}\left[\overline{\delta}^{(0)}\overline{\delta}^{(1)}\right]}. \tag{17}$$

By assumption, the estimate at the disaggregate level is unbiased, that is, $\mu^{(0)} = 0$. We let $\mu^2 = \mathbb{E}\left[\overline{\mu}^{(1)^2}\right]$ denote the expected value of the square of the bias term at the aggregate level. Using Equations 10 and 11, we may write,

$$\begin{aligned}
\mathbb{E}\left[\overline{\delta}^{(0)}\overline{\delta}^{(1)}\right] &= \frac{\sigma_\varepsilon^2}{N^{(1)}}, \\
\mathbb{E}\left[\overline{\delta}^{(0)^2}\right] &= \frac{\sigma_\varepsilon^2}{N^{(0)}}, \\
\mathbb{E}\left[\overline{\delta}^{(1)^2}\right] &= \mathbb{E}\left[\overline{\mu}^{(1)^2}\right] + \mathbb{E}\left[\overline{\varepsilon}^{(1)^2}\right] \\
&= \mu^2 + \frac{\sigma_\varepsilon^2}{N^{(1)}}.
\end{aligned}$$

These results enable us to rewrite Equation 17 for computing the weights on the disaggregate estimate using the WOPT scheme (which we denote as w^{opt}) as follows:

$$w^{opt} = \frac{1}{1+\left(\frac{1}{N^{(0)}}-\frac{1}{N^{(1)}}\right)\frac{\sigma_\varepsilon^2}{\mu^2}}. \quad (18)$$

The competing scheme, WIND, assumes independence of the estimates. The weights at the disaggregate level are obtained using the formula:

$$w^{ind} = \frac{1+\frac{1}{N^{(1)}}\frac{\sigma_\varepsilon^2}{\mu^2}}{1+\left(\frac{1}{N^{(0)}}+\frac{1}{N^{(1)}}\right)\frac{\sigma_\varepsilon^2}{\mu^2}}. \quad (19)$$

We denote by $\tilde{v}^{opt}$ and $\tilde{v}^{ind}$ the estimates computed using the two weighting schemes.

$$\begin{aligned}\tilde{v}^{opt} &= w^{opt}\bar{v}^{(0)}+(1-w^{opt})\bar{v}^{(1)},\\ \tilde{v}^{ind} &= w^{ind}\bar{v}^{(0)}+(1-w^{ind})\bar{v}^{(1)}.\end{aligned}$$

We can write the difference between the estimates of the value obtained with and without the independence assumption as $\Delta = \tilde{v}^{opt}-\tilde{v}^{ind} = \Delta_w \cdot \Delta_v$, where $\Delta_w = w^{opt}-w^{ind}$ and $\Delta_v = \bar{v}^{(0)}-\bar{v}^{(1)}$. The following proposition establishes that Δ is small under certain conditions.

Proposition 2
(i) $\lim_{\mu\to 0}\mathbb{E}[\Delta]=0$,
(ii) $\lim_{\mu\to\infty}\Delta=0$,
(iii) $\lim_{\sigma^2\to 0}\Delta=0$.

Proof:
(*i*) As $\mu\to 0$, $w^{opt}=0$ and $w^{ind}=N^{(0)}/\left(N^{(0)}+N^{(1)}\right)$. w^{ind} attains a maximum value of $1/2$ when $N^{(0)}=N^{(1)}$, but that would imply that $\bar{v}^{(0)}=\bar{v}^{(1)}\Rightarrow\Delta_v=0$. At the other extreme, if $N^{(0)}=0$, then $w^{ind}=0\Rightarrow\Delta_w=0$. For intermediate values of $N^{(0)}$, it is no longer true that the random variable Δ_v will always be zero (for statistical reasons), but we can show that its expectation will be zero using

$$\begin{aligned}\mathbb{E}[\Delta] &= \mathbb{E}\{\mathbb{E}[\Delta|\mathcal{N}]\},\\ \mathbb{E}[\Delta|\mathcal{N}] &= \mathbb{E}\left[\Delta_v\frac{N^{(0)}}{N^{(0)}+N^{(1)}}|\mathcal{N}\right]\\ &= \mathbb{E}[\Delta_v]\frac{N^{(0)}}{N^{(0)}+N^{(1)}}\\ &= 0.\end{aligned}$$

Since $\mu^2=0$, $\mathbb{E}[\Delta_v]=0$ and Equation 20 follows.
(*ii*) As $\mu\to\infty$, $w^{ind}\to w^{opt}\to 1$, which can be easily obtained by applying the appropriate limits in Equations 18 and 19. This is intuitive since with very high bias, the best strategy is to put all the weight on the most disaggregate level. As a result, $\Delta_w\to 0$.
(*iii*) As the variance goes to zero, $w^{(ind)}\to w^{(opt)}$ that again implies $\Delta_w\to 0$. □

Thus, the error from the independence assumption is small when the bias is high or low, or when the variance is low. The error will be highest for moderate values of the bias and higher values of the variance. Given that the errors vanish for the extreme cases, it is perhaps not surprising that the errors are never very large. We provide experimental evidence to support this conclusion in the next section.

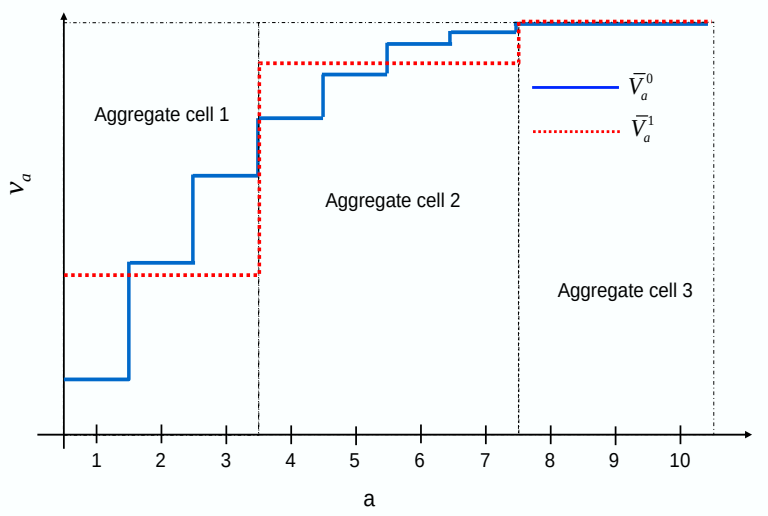


Figure 3: A piecewise constant function with its aggregate approximation. Estimates of values of each attribute vector are computed at both the aggregate and disaggregate levels. A weighted averaging is done to improve the estimates.

5.2 Experimental Results

In this section, we analyze the estimation of functions characterized by known parameters (which effectively requires that we know the actual function) in order to demonstrate the effectiveness of the optimal weighting strategy, as well as to serve as a benchmark for the strategy which assumes independent estimates at different levels of aggregation. We observe that the weights given by either method (Equations 18 and 19) are functions of the bias in the value at the aggregate level, the variance of the statistical noise in the observation of the values and the number of observations at either level. In order to compare the values of the weights from the competing strategies, we create scenarios with different combinations of the parameters that would produce significant changes in the weights. We then analyze how the variations in the weights given by WOPT and WIND affect the actual function estimates computed using the two schemes.

We consider a piecewise constant monotone function and its aggregate version as shown in Figure 3. We note that there are distinct regions in the domain where the bias is high, intermediate and zero - we expect the relative weights to be very different in these three regions. Figure 4 gives the weights (to be applied to the disaggregate level) produced by the optimal formula, WOPT, and the formula assuming independence, WIND, for each attribute a. The weights are obtained by sampling since the corresponding Equations (18 and 19) require the number of observations at the two levels of aggregation, $N^{(0)}$ and $N^{(1)}$. As we would expect, the optimal weights at the disaggregate level are zero when there is no structural error, in contrast to WIND. When the structural error is highest, the weights produced by the two methods are very similar. Note, however (consistent with our understanding from the previous section) that the weights are also quite different for the cells $a = 2$ and $a = 5$ where the aggregate and disaggregate functions are most similar (which means the bias is small). It is also the case that the weight to be given to the disaggregate level is also smallest when the bias is smallest.

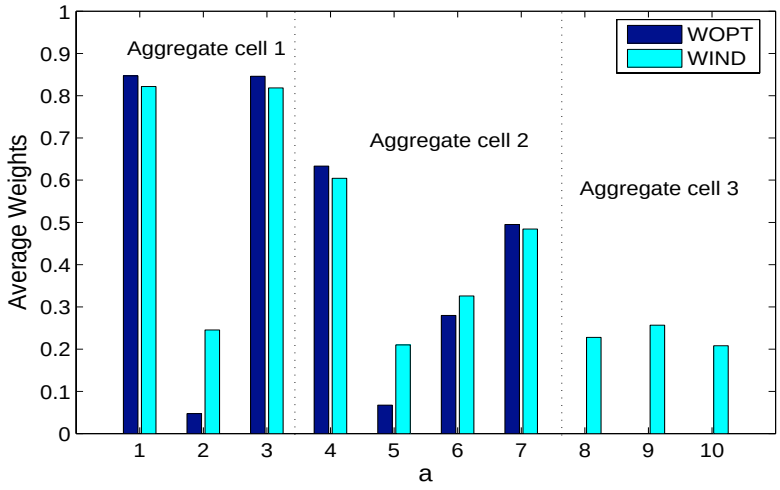


Figure 4: Comparison of the weights over the function domain

We have illustrated the difference in the weights produced by the two strategies, but less obvious is the difference in the estimates of the underlying function. In order to compare the two schemes, we developed a measure of the degree to which a weighting strategy reduced the variance of an estimate. We define the following:

$$\begin{aligned}
\tilde{v}_a^s &= \text{The value of the attribute vector } a \text{ as estimated by strategy } s.\\
\varepsilon^s &= \text{The sum of squared errors as estimated by strategy } s.\\
&= \sum_{a\in\mathcal{A}} (\tilde{v}_a^s - v_a)^2\\
\varepsilon^G &= \text{The sum of squared errors using the static aggregation strategy which treats the function as a constant over its domain.}\\
\theta^s &= \text{The performance measure for strategy } s.\\
&= 1 - \frac{\varepsilon^s}{\varepsilon^G}.
\end{aligned}$$

θ^s measures the degree of variability explained by a particular weighting strategy relative to using a single constant which can be thought of as a default strategy where all observations are aggregated together. θ^s is analogous to an R^2 measure commonly used in statistics.

A major factor in the performance of a weighting strategy is the relative size of the structural variation compared to the statistical noise. For this purpose, we define an index, ρ, that measures the ratio of the noise to the bias.

Figure 5 compares the performances of the two weighting strategies for three levels of noise. We observe that the performance of WOPT and WIND are almost identical even though there were situations where the weights given by the two schemes were significantly different. The similarity in the function estimates from the two strategies is explained by the analysis in Section 5.1.

We tested the relative performance of the two methods for other function classes. We summarize the results in figure 6 where we plot the performance measure as a function of the average number of observations per disaggregate cell. We observe that there is very little statistical difference between

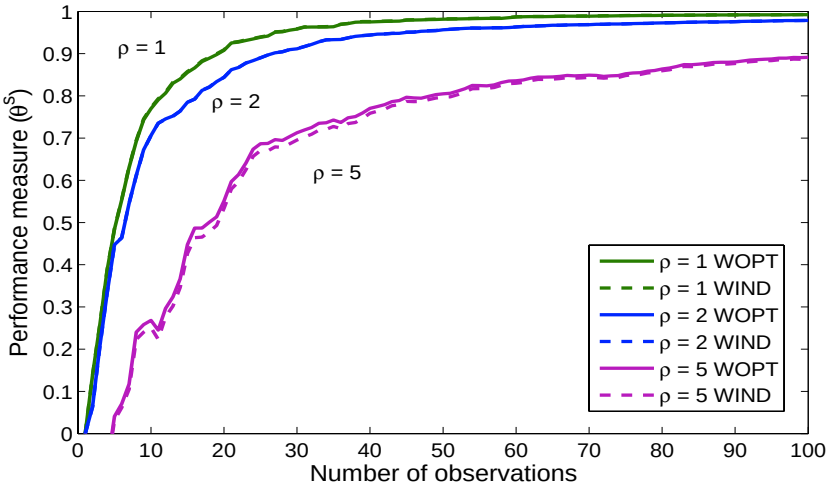


Figure 5: Comparison of the performance, as measured by θ^s, of WOPT and WIND in estimating the piecewise constant function

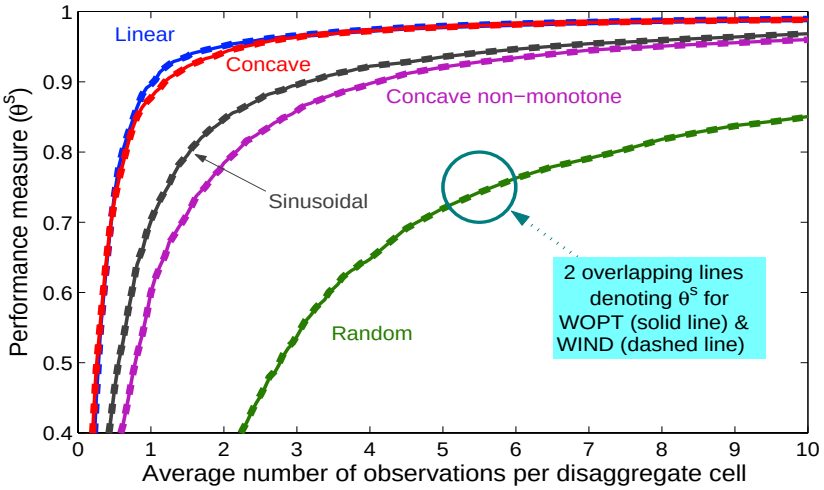


Figure 6: Comparison of WOPT and WIND for various function types using expected values of weights. The graph shows the average performance measure (θ^s) over 1000 samples for a moderate value of $\rho = 2$. WOPT is represented using solid lines and WIND, with dashed lines - the two are virtually indistinguishable.

the performance of the two methods. From this analysis, we conclude that WIND, which combines estimates assuming independence, will generally be a close approximation of WOPT. Of particular interest for our problem setting is that WIND is much easier to implement.

6. Experiments in an ADP Application

We implemented the hierarchical weighting strategy in the approximate dynamic procedure for solving the nomadic trucker problem described in Section 2. In Section 6.1, we describe the specifics of the problem instances that we consider. We also state the competing strategies that we compare in the experiments that follow. We then proceed to show the effectiveness of our hierarchical weighting scheme using two sets of experiments. In Section 6.2, we report on experiments where the discount factor is set to zero. In this case, the observations of values are unbiased, since they do not involve the estimates of values of future states. In Section 6.3, we present the results of experiments with positive discount factors. We have made available a collection of data sets used in these experiments on the following webpage - `http://castlelab.princeton.edu/`. Finally, in Section 6.4, we provide experimental results from applying our techniques on an industrial strength problem.

6.1 Experimental Design

We consider a problem where we specify the state of the truck using three attributes, namely, the current location, the day of week and the number of days away from home. The problem is rich enough to offer interesting opportunities for hierarchical aggregation, but small enough that we can solve the problem to obtain the exact solution.

The decisions are to be made over a finite time horizon of 21 time periods. The location attribute can be represented at two degrees of resolution - regions (eastern Pennsylvania, northern New Jersey) or geographical areas (Northeast, Midwest and so on). There are 50 locations at the region level which can be aggregated to 10 geographical areas.

The major contributor to the stochastic nature of the nomadic trucker problem is the uncertainty in the availability of loads in any particular location to be moved to other locations. The probability that a load is available to be moved from one location to another is dependent on the origin-destination pair. Another factor that influences the load availability is the day of week. Loads are more likely to appear during the beginning of the week (Mondays) and towards the end (Fridays). We use a probability distribution whereby the load availability dips during the middle of the week and is lower over the weekends. We introduce further uncertainty into the problem by allowing the one-period contributions to be moderately noisy.

The final attribute that we consider is the number of days that the driver is away from home. There is a penalty that we impose on moves that keep the driver away from his home domicile, which is a quadratic function of the number of days away from home. In order to keep the state space manageable (so we can obtain optimal solutions), we cap the number of days away from home at 12.

In Table 2, we list the aggregations that we use for the problem and the number of attribute states at each level of aggregation. For example, at aggregation level 1, the location attribute is aggregated from 50 regions to 10 geographical areas. We aggregate out the day-of-week attribute and retain the days-away-from-home attribute. Adding in the factor for 21 time periods, we have a total of 2541 possible states. The apparent discrepancy in the size of the state space at levels 0 and 1 arises because the days-away-from-home attribute is always set to 0 for the location corresponding to the home of the driver, while for all the other locations it can be any number from 1 to 12.

In order to compute the true values associated with each attribute vector, we use a standard backward dynamic programming algorithm. Our focus is on the problem of statistical estimation of the true values of the various states. In order to form estimates of these values we incorporate

g	Time	Location	Days-away-from-home	Day-of-week	$\lvert\mathcal{A}\rvert$
0	*	Region	*	*	86583
1	*	Area	*	-	2541
2	*	Area	-	-	210
3	*	-	-	-	21

Table 2: Aggregations for the multiattribute nomadic trucker problem. A '$*$' corresponding to a particular attribute indicates that the attribute is included in the attribute vector, and a '$-$' indicates that it is aggregated out.

our aggregation strategies in the approximate dynamic programming algorithm outlined in Figure 1. The value function estimate in Equation 1 is obtained as a weighted sum of the value estimates at various levels of aggregation:

$$\bar{v}_{a'}^{n-1} = \sum_{g\in\mathcal{G}} w_{a'}^{(g),n-1}\bar{v}_{a'}^{(g),n-1} \qquad \text{where } a' = a^M(a,d)\ \forall d \in \mathcal{D}_a$$

where $w_{a'}^{(g),n-1}$ denotes the weight, at iteration $(n-1)$, on the estimate of the value of attribute vector a' at the gth level of aggregation.

The methods that we compare use the following sets of weights:

1. Static Aggregation:

$$w_{a'}^{(g),n-1} = \begin{cases} 1 & \text{if } g \text{ is the fixed level of aggregation,} \\ 0 & \text{otherwise.} \end{cases}$$

2. Dynamic Aggregation (MINV):

$$w_{a'}^{(g),n-1} = \begin{cases} 1 & \text{if } g = \arg\min_{g'\in\mathcal{G}} \left(s_{a'}^{(g'),n-1}\right)^2 \\ 0 & \text{otherwise.} \end{cases}$$

3. WIMSE: $w_{a'}^{(g),n-1}$ is computed using Equation 16.

Equation 4 is replaced by a series of equations for updating the value function estimates at all the levels of aggregation corresponding to the currently visited attribute vector:

$$\bar{v}_a^{(g),n} = (1-\alpha)\bar{v}_a^{(g),n-1} + \alpha\hat{v}_a^n, \qquad a \in \mathcal{A}, g \in \mathcal{G}. \tag{20}$$

In the early iterations, especially for the more disaggregate levels, we often encounter the situation where the number of observations is too small (zero in certain cases) to form a meaningful estimate of a value. In such instances, we ignore the estimate at that level of aggregation. While forming a weighted sum of estimates, this usually results in all the weights being placed on the more aggregate levels.

For methods 2 and 3, we point out that the chosen level of aggregation and the weights on the estimates at different levels are dynamic in nature, in that they change with each iteration. This is brought out in the sections that follow.

The performance measure that we use for these experiments is based on the deviation of the value function approximations from the optimal value functions which can be obtained using a traditional backward dynamic programming algorithm such as value iteration. We first define

$$\begin{aligned} p_a &= \text{The steady state probability of being in state (equivalent to, in this situation, attribute vector) } a. \\ N_a &= \text{The number of observations of or visits to state } a. \\ \nu_a &= \text{The true value associated with state } a. \\ \tilde{v}_a^s &= \text{The value function approximation for state } a \text{ estimate computed using strategy } s. \end{aligned}$$

Using these we may define the following performance measures:

$$\begin{aligned} E_1^s &= \text{Error measure based on steady state probabilities, that is, a stationary distribution} \\ &= \frac{\sum_{a\in\mathcal{A}} p_a(\tilde{v}_a^s - \nu_a)}{\sum_{a\in\mathcal{A}} p_a \nu_a} \times 100\%. \qquad (21) \\ E_2^s &= \text{Error measure based on the number of visits to each state} \\ &= \frac{\sum_{a\in\mathcal{A}} N_a(\tilde{v}_a^s - \nu_a)}{\sum_{a\in\mathcal{A}} N_a \nu_a} \times 100\%. \qquad (22) \end{aligned}$$

6.2 Experiments on Myopic Data Sets

We first analyze the ability of our proposed method to make estimates from a purely statistical perspective. For this purpose, we maintain a zero discount factor, which means that the downstream values of the states are ignored while making the decisions at any time period. Instead the decisions are based on the one-period rewards alone. Using a discount factor of zero eliminates the bias that is introduced in the ADP procedure, which implies that the errors in the estimates are purely due to the noise in the observations. We compare the different estimation techniques in Table 3 where we tabulate the E_2^s values (as computed using Equation 22) for several problem instances. The static aggregation strategies for $g = 0, 1$ & 2 are denoted as *Disaggregate*, *Aggregate1* and *Aggregate2* respectively. We observe that WIMSE demonstrates lower errors than the other techniques.

6.3 Experiments on Non-myopic Data Sets

In this section, we discuss results obtained by using a positive discount factor. As such, the estimates of the future values have an impact on the current decisions. Figure 7 illustrates the variation in the weights on the estimates from different levels of aggregation, as a function of the number of observations, computed using the WIMSE weighting scheme. As expected, in the early iterations we place the highest weight on the most aggregate level since this offers the greatest statistical reliability. As the algorithm progresses, higher weights are put on the more disaggregate levels, with ultimately the highest weight on the most disaggregate level. It is interesting (and noteworthy) that the weights on the higher levels of aggregation drop quickly at first, but then stabilize, dropping

Problem #	Disaggregate		Aggregate1		MINV		WIMSE	
1	6.89	*0.07*	18.06	*0.06*	7.84	*0.13*	**5.54**	*0.08*
2	9.47	*0.07*	18.39	*0.12*	11.99	*0.28*	**7.88**	*0.09*
3	**2.92**	*0.04*	17.81	*0.07*	4.04	*0.07*	2.95	*0.04*
4	5.74	*0.08*	18.06	*0.09*	8.02	*0.16*	**5.56**	*0.07*
5	2.70	*0.05*	17.25	*0.07*	3.27	*0.11*	**2.44**	*0.05*
6	5.52	*0.09*	17.50	*0.09*	7.26	*0.10*	**5.10**	*0.10*

Table 3: Comparison of techniques for different problem instances (Disaggregate: $g = 0$, Aggregate1: $g = 1$). The performance measure used for each of the methods is the percentage deviation from optimality. Figures in italics denote the standard deviations of the terms to the left.

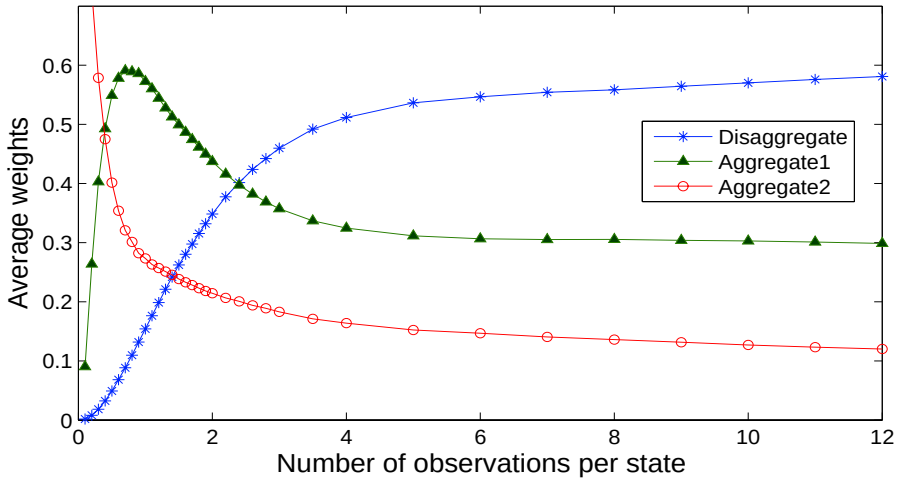


Figure 7: Average weights using hierarchical aggregation. (Disaggregate: $g = 0$, Aggregate1: $g = 1$, Aggregate2: $g = 2$).

very slowly. This behavior primarily reflects attributes with very low bias where the weight on the aggregate level will always remain fairly high.

In Figure 8, we compare the various techniques based on E_1^s values, that is, the percentage errors in the estimates from the optimal values weighted by the steady state probabilities under an optimal policy (see Equation 21). The steady state probabilities can be obtained in the process of computing the optimal values. This error measure enables us to determine how close the policy produced by the value function estimates is to an optimal policy. We see that the hierarchical weighting scheme outperforms all the static aggregation strategies as well as the dynamic aggregation strategy which picks the "best" level of aggregation, producing the least errors in the estimates.

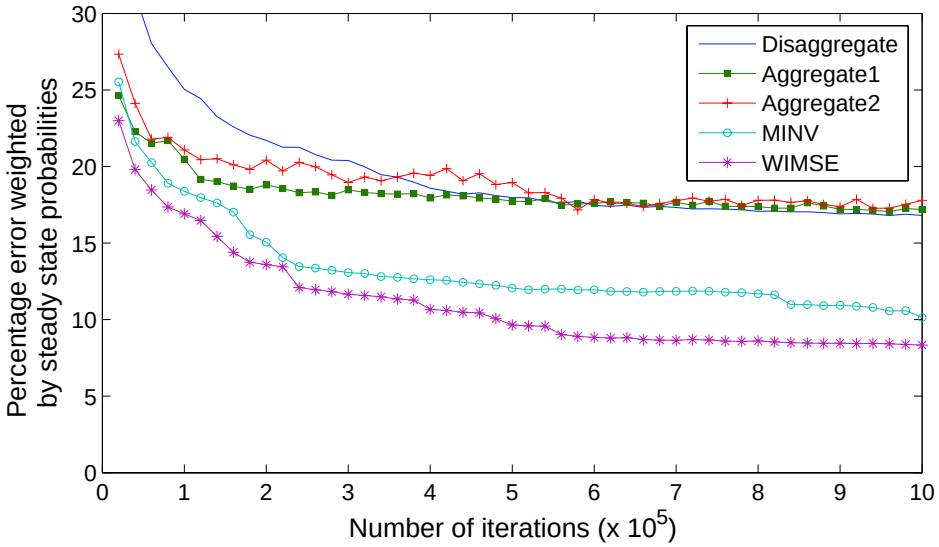


Figure 8: E_1^s values as a function of the number of observations.

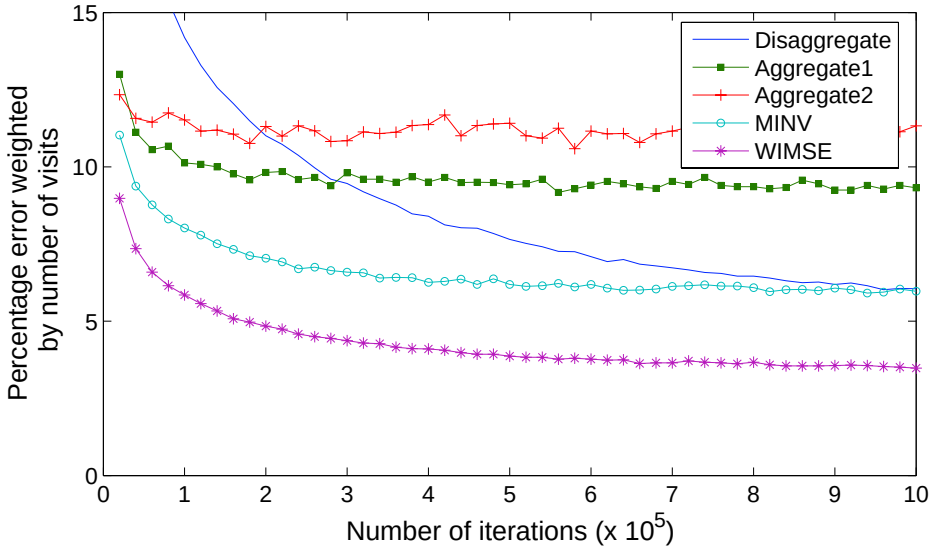


Figure 9: E_2^s values as a function of the number of observations.

Figure 9 shows the relative performance of the various schemes with respect to the errors weighted by the number of visits to the various states (E_2^s values). This error measure gives an idea of how well a particular strategy is able to estimate correctly the optimal value of the various states with higher weights on the errors in the estimates of states that are visited more frequently.

Problem #	Iterations	Disaggregate		Aggregate1		MINV		WIMSE	
1	20000	5.5	*0.1*	7.2	*0.1*	4.1	*0.0*	**2.9**	*0.0*
	50000	3.9	*0.0*	7.0	*0.1*	3.6	*0.0*	**2.2**	*0.0*
2	20000	5.3	*0.1*	6.8	*0.1*	4.8	*0.1*	**2.9**	*0.1*
	50000	3.7	*0.0*	6.6	*0.1*	4.4	*0.0*	**2.3**	*0.0*
3	20000	6.8	*0.1*	7.2	*0.1*	6.3	*0.1*	**4.3**	*0.0*
	50000	5.0	*0.0*	7.0	*0.1*	5.5	*0.1*	**3.6**	*0.0*
4	20000	11.0	*0.1*	9.8	*0.1*	7.0	*0.2*	**4.8**	*0.1*
	50000	7.7	*0.1*	9.4	*0.1*	6.2	*0.2*	**3.9**	*0.1*
5	20000	9.8	*0.1*	9.0	*0.2*	6.6	*0.1*	**5.0**	*0.1*
	50000	6.7	*0.1*	9.0	*0.1*	5.6	*0.1*	**3.8**	*0.1*
6	20000	9.9	*0.1*	8.3	*0.2*	7.1	*0.1*	**4.8**	*0.2*
	50000	6.7	*0.1*	8.2	*0.2*	6.4	*0.2*	**3.8**	*0.1*
7	20000	12.0	*0.1*	10.0	*0.2*	7.3	*0.1*	**5.7**	*0.2*
	50000	8.5	*0.1*	10.0	*0.2*	6.4	*0.1*	**4.7**	*0.1*
8	20000	11.0	*0.2*	8.6	*0.1*	7.9	*0.2*	**5.6**	*0.2*
	50000	7.8	*0.1*	8.3	*0.2*	7.2	*0.1*	**4.5**	*0.2*
9	20000	14.5	*0.4*	13.4	*0.3*	10.6	*0.5*	**8.4**	*0.3*
	50000	10.6	*0.4*	12.5	*0.3*	9.4	*0.4*	**7.3**	*0.3*
10	20000	13.8	*0.2*	13.1	*0.5*	10.6	*0.2*	**8.3**	*0.3*
	50000	9.7	*0.2*	12.7	*0.2*	9.0	*0.2*	**6.8**	*0.2*

Table 4: Comparison of techniques for different instances of the nomadic trucker problem. Two higher levels of aggregation are also used in the problem, but omitted from the tables, as they give inferior results. Figures in italics denote the standard deviations of the terms to the left.

Here again, WIMSE is found to consistently outperform the remaining techniques, producing error values that are much lower.

We now compare the various techniques for several problem instances. The major parameters that are varied to obtain the different problem sets are the discount factor, the probability distribution of the load availability at the various locations and the level of uncertainty in the contributions resulting from choosing a particular decision. We used discount factors of 0.80, 0.90 and 0.95, three different load probability distributions and two levels of uncertainty in the rewards. The E_2^s values for these problem instances are tabulated in Table 4. Here again, WIMSE outperforms all the other schemes consistently, irrespective of the number of iterations.

6.4 Experiments with a Fleet of Trucks

The techniques in this paper were designed to be applied to approximate dynamic programming algorithms for optimizing a fleet of trucks. To illustrate, we provide a very brief model of a multiattribute resource allocation problem that can be applied to the management of large fleets of trucks, freight cars, locomotives, military air cargo jets or business jets (we have industrial experience with

all of these problems). Let

$$\begin{aligned}
R_a &= \text{The number of resources with attribute } a.\\
R &= (R_a)_{a\in\mathcal{A}}\\
\mathcal{D} &= \text{The set of different decision types that can be used to act on a resource (e.g., moving from one location to another, repairing a vehicle).}\\
x_{ad} &= \text{The number of times we act on a resource with attribute } a \text{ using a decision of type } d\in\mathcal{D}.\\
x &= (x_{ad})_{a\in\mathcal{A},d\in\mathcal{D}}\\
c_{ad} &= \text{The contribution generated by } x_{ad}.
\end{aligned}$$

For problems with moderately complex attribute vectors, we have found that we can obtain high quality solutions using dynamic programming approximations by solving (at iteration n) approximations of the form,

$$\tilde{V}^n(R^n) = \max_x \left(\sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} c_{ad}x_{ad} + \sum_{a'\in\mathcal{A}} \bar{v}^{n-1}_{a'} R^x_{a'}(x) \right) \tag{23}$$

subject to,

$$\sum_{d\in\mathcal{D}} x_{ad} = R^n_a \quad \forall\, a\in\mathcal{A}, \tag{24}$$

$$R^x_{a'} = \sum_{a\in\mathcal{A}} \sum_{d\in\mathcal{D}} \delta_{a'} x_{ad}, \tag{25}$$

$$x_{ad} \geq 0. \tag{26}$$

This is a fairly basic model, but it is enough to illustrate the application (for a more complete description, see Spivey and Powell 2004). Problem 23 - 26 is a linear program, and returns a dual variable that we denote $\hat{v}^n_a$ for the resource constraint in Equation 24. In a real application, the attribute space $\mathcal{A}$ is large enough that we do not generate Equation 24 for each $a\in\mathcal{A}$. Instead, we might generate the equation only if $R^n_a > 0$ (for example). We can then update our value function approximation using

$$\bar{v}^n_a = (1-\alpha_n)\bar{v}^{n-1}_a + \alpha_n \hat{v}^n_a.$$

The problem we encounter, just as in the earlier sections, is that real problems might have attribute spaces with several million elements, and yet we can only run a few hundred iterations of the algorithm. Many of the attributes are never sampled, while others will have only a few observations. A small handful may receive dozens of observations. As a result, the statistical reliability of the approximations $\bar{v}^n_a$ can be quite low. The standard technique is to estimate the values at some aggregate level that trades off between the statistical reliability and cost of bias introduced by aggregation. The right level of aggregation depends not only on the specific attribute (some are sampled more often than others) but also on the number of iterations we have run the algorithm.

We tested our multilevel weighting strategy using this model to simulate the operations of a major truckload motor carrier. A detailed description of this problem can be found in Simao et al. (2008). The most disaggregate attribute vector used for the value function captured the location

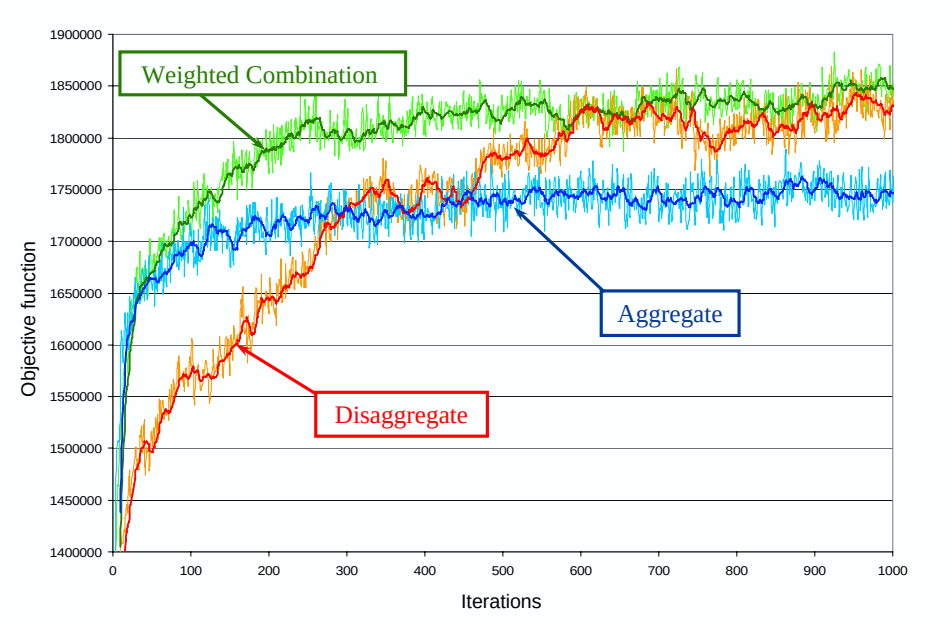


Figure 10: Comparison of using a single level of aggregation (aggregate and disaggregate) against a weighted aggregation strategy for optimizing a fleet of trucks.

of the driver (out of 100 regions), the driver's home domicile (out of 100 regions), and whether the driver was a single driver employed by the company, a contract driver, or a team (two drivers working together). At the most disaggregate level, the attribute space consisted of 30,000 elements. We considered an aggregate level which captured only the location of the driver (100 elements) and a hierarchical strategy that used a weighted estimate across four levels of aggregation.

The results of the experiment are shown in Figure 10. For this problem, a single iteration of the ADP algorithm can take 20 minutes. Even if we had confidence that the solution quality would never be better, improving the rate of convergence is extremely important. Since it is required to run this model continually to perform policy analyses, it would be extremely advantageous if we could reduce the number of iterations needed for convergence from 1000 to 200. As expected, if we use only an aggregate attribute vector, we obtain fast convergence but it levels off quickly. Using the most disaggregate representation produces slow convergence but it eventually reaches a better solution. By contrast, using a weighted combination of estimates at different levels of aggregation produced the fastest convergence and the best results overall. We point out that faster convergence to a better solution does not necessarily imply gains in running time. Even though the weighted combination strategy has a greater number of computations per iteration compared to static aggregation schemes, we have observed that the increase in running time for additional computations is not significant.

7. Conclusion

There is a vast range of problems that can be described as multiattribute resource allocation problems that can be modeled as stochastic dynamic programs. All of these problems pose the statistical challenge of estimating the value of a resource as a function of a potentially complex vector of attributes. Requiring no more than a set of aggregation functions which are typically quite easy to design, we have proposed an adaptive weighting strategy that produces an estimate of the value of a resource with a particular attribute. The weighting system is quite easy to compute, and is implementable for very large scale problems with attribute spaces that are arbitrarily large, since the computational complexity is a function of how many attributes you actually visit. The weights adjust naturally as the number of observations change. Thus, it performs well in the early iterations when there are very few observations. As more observations are made of particular attributes, the weight given to disaggregate estimates increases, producing higher quality solutions.

Our most surprising result was the finding that a weighting system that assumed independence of the estimates at each level of aggregation worked so well. This assumption is clearly not accurate. However, our analysis showed that it had little or no effect on the accuracy of a prediction. If the difference between a disaggregate estimate and the corresponding aggregate estimate was large, the weights produced by our approximation closely matched the weights that would have been produced had we properly accounted for the correlation. If the difference between the disaggregate and aggregate estimates was small, the weights could be quite different, but in this case it did not matter. The result is a weighting system that is easy to compute, scales well to very large scale applications and provides very accurate estimates.

An interesting possibility for future work would be to analytically solve for the optimal weights to combine estimates when there are more than two levels of aggregation. Further, the heuristic techniques that we have proposed could be applied to a broader range of estimation problems, not restricted to applications in approximate dynamic programming.

Acknowledgments

The authors would like to recognize the helpful comments of several referees. This research was supported in part by grant AFOSR contract FA9550-08-1-0195, and by grant N00014-07-1-0150 from the Office of Naval Research through the Center for Dynamic Data Analysis.

Appendix A. Proof of Proposition 1:

We first prove (8). The Lagrangian for the problem formulated in (6)-(7) is

$$\begin{aligned} L(w,\lambda) &= \mathbb{E}\left[\frac{1}{2}\left(\sum_{g\in\mathcal{G}} w_a^{(g)}\cdot\bar{v}_a^{(g)} - \nu_a\right)^2\right] + \lambda\left(1-\sum_{g\in\mathcal{G}} w_a^{(g)}\right) \\ &= \mathbb{E}\left[\frac{1}{2}\left(\sum_{g\in\mathcal{G}} w_a^{(g)}\left(\bar{v}_a^{(g)} - \nu_a\right)\right)^2\right] + \lambda\left(1-\sum_{g\in\mathcal{G}} w_a^{(g)}\right). \end{aligned}$$

The first order optimality conditions are easily shown to be

$$\begin{aligned}\mathbb{E}\left[\sum_{g\in\mathcal{G}} w_a^{(g)}\left(\overline{v}_a^{(g)}-\nu_a\right)\left(\overline{v}_a^{(g')}-\nu_a\right)\right]-\lambda &= 0 \quad \forall\ g'\in\mathcal{G}, \\ \sum_{g\in\mathcal{G}} w_a^{(g)}-1 &= 0.\end{aligned} \tag{27}$$

To simplify Equation 27, we note that,

$$\begin{aligned}\mathbb{E}\left[\sum_{g\in\mathcal{G}} w_a^{(g)}\left(\overline{v}_a^{(g)}-\nu_a\right)\left(\overline{v}_a^{(g')}-\nu_a\right)\right] &= \mathbb{E}\left[\sum_{g\in\mathcal{G}} w_a^{(g)}\overline{\delta}_a^{(g)}\overline{\delta}_a^{(g')}\right] \\ &= \sum_{g\in\mathcal{G}} w_a^{(g)}\mathbb{E}\left[\overline{\delta}_a^{(g)}\overline{\delta}_a^{(g')}\right].\end{aligned} \tag{28}$$

Combining Equations 27 and 28 gives us Equation 8.

We now derive Equation 10. We assume that the bias error, $\overline{\mu}_a^{(g)}$, is uncorrelated with the random error, $\overline{\varepsilon}_a^{(g)}$. Using Equation 5, we obtain the relation,

$$\begin{aligned}\mathbb{E}\left[\overline{\delta}_a^{(g)}\overline{\delta}_a^{(g')}\right] &= \mathbb{E}\left[(\overline{\mu}_a^{(g)}+\overline{\varepsilon}_a^{(g)})(\overline{\mu}_a^{(g')}+\overline{\varepsilon}_a^{(g')})\right] \\ &= \mathbb{E}\left[\overline{\mu}_a^{(g)}\overline{\mu}_a^{(g')}+\overline{\mu}_a^{(g')}\overline{\varepsilon}_a^{(g)}+\overline{\mu}_a^{(g)}\overline{\varepsilon}_a^{(g')}+\overline{\varepsilon}_a^{(g)}\overline{\varepsilon}_a^{(g')}\right] \\ &= \mathbb{E}\left[\overline{\mu}_a^{(g)}\overline{\mu}_a^{(g')}\right]+\mathbb{E}\left[\overline{\mu}_a^{(g')}\overline{\varepsilon}_a^{(g)}\right]+\mathbb{E}\left[\overline{\mu}_a^{(g)}\overline{\varepsilon}_a^{(g')}\right]+\mathbb{E}\left[\overline{\varepsilon}_a^{(g)}\overline{\varepsilon}_a^{(g')}\right].\end{aligned} \tag{29}$$

We notice that, $\mathbb{E}\left[\overline{\mu}_a^{(g')}\overline{\varepsilon}_a^{(g)}\right]=\mathbb{E}\left[\overline{\mu}_a^{(g')}\right]\mathbb{E}\left[\overline{\varepsilon}_a^{(g)}\right]=0$. By a similar argument, $\mathbb{E}\left[\overline{\mu}_a^{(g)}\overline{\varepsilon}_a^{(g')}\right]=0$. This enables us to rewrite Equation 29 as,

$$\mathbb{E}\left[\overline{\delta}_a^{(g)}\overline{\delta}_a^{(g')}\right] = \mathbb{E}\left[\overline{\mu}_a^{(g)}\overline{\mu}_a^{(g')}\right]+\mathbb{E}\left[\overline{\varepsilon}_a^{(g)}\overline{\varepsilon}_a^{(g')}\right]. \tag{30}$$

Since $g'>g$, the stochastic error term at level g', $\overline{\varepsilon}_a^{(g')}$, can be expressed as a combination of $\overline{\varepsilon}_a^{(g)}$ and some terms that are independent of it:

$$\begin{aligned}\overline{\varepsilon}_a^{(g')} &= \frac{1}{N_a^{(g')}}\sum_{n\in\mathcal{N}_a^{(g')}}\varepsilon^n \\ &= \frac{1}{N_a^{(g')}}\left(\sum_{n\in\mathcal{N}_a^{(g)}}\varepsilon^n+\sum_{n\in\mathcal{N}_a^{(g')}\setminus\mathcal{N}_a^{(g)}}\varepsilon^n\right) \\ &= \frac{N_a^{(g)}}{N_a^{(g')}}\overline{\varepsilon}_a^{(g)}+\frac{1}{N_a^{(g')}}\sum_{n\in\mathcal{N}_a^{(g')}\setminus\mathcal{N}_a^{(g)}}\varepsilon^n.\end{aligned} \tag{31}$$

Using Equation 31, we can rewrite the second term on the right hand side of Equation 30 term as follows:

$$\mathbb{E}\left[\overline{\varepsilon}_a^{(g)}\overline{\varepsilon}_a^{(g')}\right] = \mathbb{E}\left[\overline{\varepsilon}_a^{(g)}\cdot\frac{N_a^{(g)}}{N_a^{(g')}}\overline{\varepsilon}_a^{(g)}\right]+\mathbb{E}\left[\overline{\varepsilon}_a^{(g)}\cdot\frac{1}{N_a^{(g')}}\sum_{n\in\mathcal{N}_a^{(g')}\setminus\mathcal{N}_a^{(g)}}\varepsilon^n\right]$$

$$= \frac{N_a^{(g)}}{N_a^{(g')}} \mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \bar{\varepsilon}_a^{(g)}\right] + \frac{1}{N_a^{(g')}} \underbrace{\mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \cdot \sum_{n \in \mathcal{N}_a^{(g')} \setminus \mathcal{N}_a^{(g)}} \varepsilon^n\right]}_{I}.$$

Since the individual observations are assumed to be independent, the term I can be further simplified as follows,

$$\begin{aligned}
\mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \cdot \sum_{n \in \mathcal{N}_a^{(g')} \setminus \mathcal{N}_a^{(g)}} \varepsilon^n\right] &= \mathbb{E}\left[\bar{\varepsilon}_a^{(g)}\right] \mathbb{E}\left[\sum_{n \in \mathcal{N}_a^{(g')} \setminus \mathcal{N}_a^{(g)}} \varepsilon^n\right] \\
&= 0.
\end{aligned}$$

This implies that,

$$\begin{aligned}
\mathbb{E}\left[\bar{\varepsilon}_a^{(g)} \bar{\varepsilon}_a^{(g')}\right] &= \frac{N_a^{(g)}}{N_a^{(g')}} \mathbb{E}\left[\bar{\varepsilon}_a^{(g)^2}\right] \\
&= \frac{N_a^{(g)}}{N_a^{(g')}} \mathbf{Var}\left[\bar{\varepsilon}_a^{(g)}\right] \\
&= \frac{N_a^{(g)}}{N_a^{(g')}} \frac{1}{N_a^{(g)}} \mathbf{Var}\left[\varepsilon^n\right] \\
&= \frac{1}{N_a^{(g')}} \sigma_\varepsilon^2. \qquad (32)
\end{aligned}$$

Combining Equations (30 and 32 gives us the result in Equation 10. This results generalizes for all $g \in \mathcal{G}$, since g and g' can be interchanged when $g > g'$.

References

M. Athans, D. Bertsekas, W. McDermott, J. Tsitsiklis, and B. Van Roy. Intelligent optimal control. Technical report, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 1995.

J.C. Bean, J.R. Birge, and R.L. Smith. Aggregation in dynamic programming. *Operations Research*, 35:215–220, 1987.

D. Bertsekas and D. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989.

D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *J. of Artificial Intelligence*, 11:1–94, 1999.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000. URL `citeseer.ist.psu.edu/boutilier99stochastic.html`.

L. Breiman. Stacked regression. *Machine Learning*, 24:49–64, 1996.

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

Z. Feng, E. A. Hansen, and S. Zilberstein. Symbolic generalization for on-line planning. In Christopher Meek and Uffe Kjærulff, editors, *UAI*, pages 209–216. Morgan Kaufmann, 2003. ISBN 0-127-05664-5.

M. Gendreau and J. Y. Potvin. Dynamic vehicle routing and dispatching. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 115–126. Kluwer Academic Publishers, 1998.

I. Guttman, S.S. Wilks, and J.S. Hunter. *Introductory Engineering Statistics*. John Wiley and Sons, Inc., New York, NY, 1965.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer series in Statistics, New York, NY, 2001.

S. Ichoua, M. Gendreau, and J.-Y. Potvin. Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225, 2005.

K.E. Kim and T. Dean. Solving factored MDP's using non-homogeneous partitions. *Artificial Intelligence*, 147(1-2):225–251, 2003.

M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. *Journal of the American Statistical Association*, 91:1641–1650, 1996.

R. Luus. *Iterative Dynamic Programming*. Chapman & Hall/CRC, New York, 2000.

R. Mendelssohn. An iterative aggregation procedure for Markov decision processes. *Operations Research*, 30(1):62–73, 1982.

W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley and Sons, New York, 2007.

W. B. Powell and T. A. Carvalho. Dynamic control of logistics queueing networks for large-scale fleet management. *Transportation Science*, 32(2):90–109, 1998.

W. B. Powell, J. A. Shapiro, and H. P. Simão. An adaptive dynamic programming algorithm for the heterogeneous resource allocation problem. *Transportation Science*, 36(2):231–249, 2002.

D. Rogers, R. Plante, R. Wong, and J. Evans. Aggregation and disaggregation techniques and methodology in optimization. *Operations Research*, 39(4):553–582, 1991.

N. Secomandi. Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Computers and Operations Research*, 27(11):1201–1225, 2000.

N. Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

H. P. Simao, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell. An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, (to appear), 2008.

M. Spivey and W. B. Powell. The dynamic assignment problem. *Transportation Science*, 38(4): 399–419, 2004.

R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3: 9–44, 1988.

R.S. Sutton and A.G. Barto. *Reinforcement Learning*. The MIT Press, Cambridge, Massachusetts, 1998.

J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

X. Wang and T. G. Dietterich. Efficient value function approximation using regression trees. In J. Boyan, W. Buntine, and A. Jagota, editors, *Statistical Machine Learning for Large Scale Optimization, Neural Computing Surveys*. 2000.

C.J.C.H. Watkins. Learning from delayed rewards. Ph.d. thesis, Cambridge University, Cambridge, UK, 1989.

W. Whitt. Approximations of dynamic programs I. *Mathematics of Operations Research*, 3:231–243, 1978.

D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

Y. Yang. Adaptive regression by mixing. *Journal of the American Statistical Association*, 96, 2001.

Q. Zhang and S. P. Sethi. Near optimization of dynamic systems by decomposition and aggregation. *Journal of Optimization Theory and Applications*, 99(1):1–22, 1998.

Journal of Machine Learning Research 9 (2008) 2113-2139 Submitted 1/08; Published 10/08

Gradient Tree Boosting for Training Conditional Random Fields

Thomas G. Dietterich TGD@EECS.OREGONSTATE.EDU
Guohua Hao HAOG@EECS.OREGONSTATE.EDU
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331, USA

Adam Ashenfelter ASHENFAD@CLEVERSET.COM
Cleverset, Inc.
Corvallis, OR 97330, USA

Editor: Michael Collins

Abstract

Conditional random fields (CRFs) provide a flexible and powerful model for sequence labeling problems. However, existing learning algorithms are slow, particularly in problems with large numbers of potential input features and feature combinations. This paper describes a new algorithm for training CRFs via gradient tree boosting. In tree boosting, the CRF potential functions are represented as weighted sums of regression trees, which provide compact representations of feature interactions. So the algorithm does not explicitly consider the potentially large parameter space. As a result, gradient tree boosting scales linearly in the order of the Markov model and in the order of the feature interactions, rather than exponentially as in previous algorithms based on iterative scaling and gradient descent. Gradient tree boosting also makes it possible to use instance weighting (as in C4.5) and surrogate splitting (as in CART) to handle missing values. Experimental studies of the effectiveness of these two methods (as well as standard imputation and indicator feature methods) show that instance weighting is the best method in most cases when feature values are missing at random.

Keywords: sequential supervised learning, conditional random fields, functional gradient, gradient tree boosting, missing values

1. Introduction

Many applications of machine learning involve assigning labels collectively to sequences of objects. For example, in natural language processing, the task of part-of-speech (POS) tagging is to label each word in a sentence with a part of speech tag ("noun", "verb" etc.) (Ratnaparkhi, 1996). In computational biology, the task of protein secondary structure prediction is to assign a secondary structure class to each amino acid residue in the protein sequence (Qian and Sejnowski, 1988).

We call this class of problems *sequential supervised learning* (SSL), and it can be formulated as follows:

Given: A set of training examples of the form (X_i, Y_i), where each $X_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,T_i})$ is a sequence of T_i feature vectors and each $Y_i = (y_{i,1}, \ldots, y_{i,T_i})$ is a corresponding sequence of class labels, where $y_{i,t} \in \{1, \ldots, K\}$.

Find: A classifier H that, given a new sequence X of feature vectors, predicts the corresponding sequence of class labels $Y = H(X)$ accurately.

Perhaps the most famous SSL problem is the NETtalk task of pronouncing English words by assigning a phoneme and stress to each letter of the word (Sejnowski and Rosenberg, 1987). Other applications of SSL arise in information extraction (McCallum et al., 2000) and handwritten word recognition (Taskar et al., 2004).

Early attempts to apply machine learning to SSL problems were based on *sliding windows*. To predict label y_t, a sliding window method uses features drawn from some "window" of the X sequence. For example, a 5-element window $w_t(X)$ would use the features $\mathbf{x}_{t-2}, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}$. Sliding windows convert the SSL problem into a standard supervised learning problem to which any ordinary machine learning algorithm can be applied. However, in most SSL problems, there are correlations among successive class labels y_t. For example, in part-of-speech tagging, adjectives tend to be followed by nouns. In protein sequences, alpha helixes and beta structures always involve multiple adjacent residues. These correlations can be exploited to increase classification accuracy.

The best-known method for capturing the $y_{t-1} \leftrightarrow y_t$ correlation is the hidden Markov model (HMM) (Rabiner, 1989), which is a generative model of $P(X,Y)$, the joint distribution of the observation sequence and label sequence. In this model, the joint distribution is factored as $P(X,Y) = \prod_t P(y_t|y_{t-1})P(\mathbf{x}_t|y_t)$, and the observation distribution is further factored as $P(\mathbf{x}_t|y_t) = \prod_j P(\mathbf{x}_{t,j}|y_t)$. This assumption of independence of each input feature $\mathbf{x}_{t,j}$ conditioned on y_t makes HMMs unable to model arbitrary, non-independent input features, and this limits the accuracy and "engineerability" of HMMs.

Recent research has instead focused on discriminative models, in which arbitrary and non-independent observation features can be easily incorporated. Much machine learning research has shown that discriminative models tend to be more accurate and more robust to incorrect modeling assumptions (Ng and Jordan, 2002). McCallum and his collaborators introduced maximum entropy Markov models (MEMMs) (McCallum et al., 2000) and conditional random fields (CRFs) (Lafferty et al., 2001). MEMMs are directed graphical models of the form $P(Y|X) = \prod_t P(y_t|y_{t-1}, w_t(X))$, where $w_t(X)$ is a sliding window over the X sequence. They are easy to train, but they suffer from the *label bias problem* that results from the local normalization at each time step t. Conditional random fields are undirected models of the form $P(Y|X) = 1/Z(X) \exp \sum_t \Psi(y_t, y_{t-1}, w_t(X))$, where $Z(X)$ is a global normalizing term and $\Psi(y_t, y_{t-1}, w_t(X))$ is a potential function that scores the compatibility of y_t, y_{t-1}, and $w_t(X)$. The global normalization avoids the label bias problem but makes training much more computationally expensive. CRFs have been applied to many problems with excellent results including POS tagging (Lafferty et al., 2001) and noun-phrase chunking (Sha and Pereira, 2003).

Kernel-based methods have also been extended to the SSL case. The hidden Markov SVM (Altun et al., 2003; Tsochantaridis et al., 2004) and max-margin Markov networks (Taskar et al., 2004) learn a discriminant function $F(X,Y')$ that assigns a real valued score to each possible label sequence Y' to maximize the margin between the correct label sequence Y and all competing incorrect label sequences.

Training CRFs is difficult for several reasons. First, as with all collective classification problems, training requires performing inference. In particular, all algorithms must compute the conditional log likelihood $\log P(Y_i|X_i)$ for each training example (X_i, Y_i) in each iteration. This is expensive, and it dictates that training algorithms should try to minimize the number of iterations and maximize the amount of progress made in each iteration. Second, in many SSL applications, the space of potential

features for describing the arguments of ψ (i.e., y_t, y_{t-1}, and $w_t(X)$) is immense. Even in the simple case where ψ is represented as a simple linear function $W \cdot F(y_t, y_{t-1}, w_t(X))$, there can be millions of weights to learn in W. In POS tagging and semantic role labeling, for example, it is common to have one feature (and hence, one weight) for every combination of a word and a pair of class labels. Furthermore, in most applications, performance is improved if the algorithm can consider combinations of these basic features (e.g., word n-grams, feature conjunctions and disjunctions, etc.). If feature interactions are permitted, the number of parameters to be learned explodes. Finally, in some problems, feature values can be missing, and this is difficult for discriminative training algorithms to handle.

There has been steady progress in algorithms for training CRFs. The initial paper (Lafferty et al., 2001) introduced an iterative scaling algorithm, which was reported to be exceedingly slow. Several groups have implemented gradient ascent methods (such as Sha and Pereira, 2003), but naive implementations are also very slow. McCallum's Mallet system (McCallum, 2002) employs the BFGS algorithm, which is an approximate second order method, to speed up the training of CRFs and improve the prediction accuracy. More recently, Vishwanathan et al. (2006) proposed to use stochastic gradient method to train CRFs, and accelerate this process via the Stochastic Meta-Descent (SMD), which is a gain adaptation method. The resulting algorithm is much faster than the BFGS algorithm and scales well on large data sets.

In this paper, we introduce a different approach for training the potential functions based on Friedman's gradient tree boosting algorithm (Friedman, 2001). In this method, the potential functions are represented by sums of regression trees, which are grown stage-wise in the manner of Adaboost (Freund and Schapire, 1996). Because each iteration adds an entire regression tree to the potential function, each iteration can take a big step in parameter space, and hence, reduce the number of iterations needed. Tree boosting also addresses the problem of dealing with feature interactions. Each regression tree can be viewed as defining several new feature combinations—one corresponding to each path in the tree from the root to a leaf. The resulting potential functions still have the form of a linear combination of features, but the features can be quite complex. Another advantage of tree boosting is that it is able to handle missing values in the inputs using clever methods specific to regression trees, such as the instance weighting method of C4.5 (Quinlan, 1993) and the surrogate splitting method of CART (Breiman et al., 1984). Finally, the algorithm is fast and straightforward to implement. In addition, there may be some tendency to avoid overfitting because of the "ensemble effect" of combining multiple regression trees.

This paper describes the gradient tree boosting algorithm including methods for incorporating weight penalties into the procedure. It then compares training time and generalization performance against McCallum's Mallet system. The results show that our implementation of tree boosting is competitive with Mallet in both speed and accuracy and that additional improvements in our implementation of the forward-backward algorithm would likely produce a system that is faster than both systems. We also perform experiments to evaluate the effectiveness of four methods for handling missing values (instance weighting, surrogate splits, indicator features, and imputation). The results show that instance weighting works best, but that imputation also works surprisingly well.

This leads to two conclusions. First, for CRF models, instance weighting combined with gradient tree boosting can be recommended as a good algorithm for learning in the presence of missing values. Second, for all SSL methods, imputation can be employed to provide a reasonable missing values method.

2. Conditional Random Fields

Let (X,Y) be a sequential labeled training example, where $X = (\mathbf{x}_1,\ldots,\mathbf{x}_T)$ is the observation sequence and $Y = (y_1,\ldots,y_T)$ is the sequence of labels, where $y_t \in \{1,\ldots,K\}$ for all t. A conditional random field is a linear chain Markov random field (Geman and Geman, 1984) over the label sequence Y globally conditioned on the observation sequence X. The probability distribution can be written as

$$P(Y|X) = \frac{1}{Z(X)} \exp\left[\sum_t \Psi_t(y_t,X) + \Psi_{t-1,t}(y_{t-1},y_t,X)\right] \ ,$$

where $\Psi_t(y_t,X)$ and $\Psi_{t-1,t}(y_{t-1},y_t,X)$ are *potential functions* that capture (respectively) the degree to which y_t is compatible with X and the degree to which y_t is compatible with a transition from y_{t-1} and with X. These potential functions can be arbitrary real-valued functions. The exponential function ensures that $P(Y|X)$ is positive, and the normalizing constant $Z(X) = \sum_{Y'} \exp[\sum_t \Psi_t(y'_t,X) + \Psi_{t-1,t}(y'_{t-1},y'_t,X)]$ ensures that $P(Y|X)$ sums to 1. If given sufficiently rich potential functions, this model can represent any first-order Markov distribution $P(Y|X)$ subject to the assumption that $P(Y|X) > 0$ for all X and Y (Besag, 1974; Hammersley and Clifford, 1971). Normally, it is assumed that the potential functions do not depend on t, and we will adopt this assumption in this paper.

To apply a CRF to an SSL problem, we must choose a representation for the potential functions. Lafferty et al. (2001) studied potential functions that are weighted combinations of binary features:

$$\begin{aligned} \Psi_t(y_t,X) &= \sum_a \beta_a g_a(y_t,X) \ , \\ \Psi_{t-1,t}(y_{t-1},y_t,X) &= \sum_b \lambda_b f_b(y_{t-1},y_t,X) \ , \end{aligned}$$

where the β_a's and λ_b's are trainable weights, and the features g_a and f_b are boolean functions. In part-of-speech tagging, for example, $g_{234}(y_t,X)$ might be 1 when $\mathbf{x}_t$ is the word "bank" and y_t is the class "noun" (and 0 otherwise). As with sliding window methods, it is natural to define features that depend only on a sliding window $w_t(X)$ of X values. This linear parameterization can be seen as an extension of logistic regression to the sequential case.

CRFs can be trained by maximizing the log likelihood of the training data, possibly with a regularization penalty to prevent overfitting. Let $\Theta = \{\beta_1,\ldots,\lambda_1,\ldots\}$ denote all of the tunable parameters in the model. Then we seek to maximize the objective function

$$\begin{aligned} J(\Theta) &= \log \prod_i P(Y_i \mid X_i) \\ &= \sum_i \log \frac{1}{Z(X_i)} \exp\left[\sum_t \Psi_t(y_{i,t},X_i) + \Psi_{t-1,t}(y_{i,t-1},y_{i,t},X_i)\right] \\ &= \sum_i \sum_t \Psi_t(y_{i,t},X_i) + \Psi_{t-1,t}(y_{i,t-1},y_{i,t},X_i) - \log Z(X_i) \\ &= \sum_i \sum_t \sum_a \beta_a g_a(y_{i,t},X_i) + \sum_b \lambda_b f_b(y_{i,t-1},y_{i,t},X_i) - \log Z(X_i) \ . \end{aligned}$$

A drawback of this linear parameterization is that it assumes that each feature makes an independent contribution to the potential functions. Of course it is possible to define more features to capture combinations of the basic features, but this leads to a combinatorial explosion in the number

of features, and hence, in the dimensionality of the optimization problem. For example, in protein secondary structure prediction, Qian and Sejnowski (1988) found that a 13-residue sliding window gave best results for neural network methods. There are $3^2 \times 13 \times 20 = 2340$ basic f_b features that can be defined over this window. If we consider fourth-order conjunctions of such features, we obtain more than 10^{12} features. This is obviously infeasible.

McCallum's Mallet system (McCallum, 2002) implements standard CRFs and CRFs with feature induction (McCallum, 2003). When feature induction is turned on, the learner starts with a single constant feature and (every 8 iterations) introduces new feature conjunctions by taking conjunctions of the basic features with features already in the model. Candidate conjunctions are evaluated according to their incremental impact on the objective function. He demonstrates significant improvements in speed and classification accuracy compared to a CRF that only includes the basic features. In this paper, we employ the gradient tree boosting method (Friedman, 2001) to construct complex features from the basic features as part of a stage-wise construction of the potential functions. The regression trees grown at each step are compact representations of complex features.

3. Gradient Tree Boosting

Suppose we wish to solve a standard supervised learning problem where the training examples have the form $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$ and $y_i \in \{1, \ldots, K\}$. We wish to fit a model of the form

$$P(y \mid \mathbf{x}) = \frac{\exp \Psi(y, \mathbf{x})}{\sum_{y'} \exp \Psi(y', \mathbf{x})} \ .$$

Gradient tree boosting is based on the idea of *functional gradient ascent*. In ordinary gradient ascent, we would parameterize Ψ in some way, for example, as a linear function,

$$\Psi(y, \mathbf{x}) = \sum_a \beta_a g_a(y, \mathbf{x}) \ .$$

Let $\Theta = \{\beta_1, \ldots\}$ represent all of the tunable parameters in this function. In gradient ascent, the fitted parameter vector after iteration m, Θ_m, is a sum of an initial parameter vector Θ_0 and a series of gradient ascent steps δ_m:

$$\Theta_m = \Theta_0 + \delta_1 + \cdots + \delta_m \ ,$$

where each δ_m is computed as a step in the direction of the gradient of the log likelihood function:

$$\delta_m = \eta_m \nabla_\Theta \sum_i \log P(y_i \mid \mathbf{x}_i; \Theta) \Big|_{\Theta_{m-1}} \ ,$$

and η_m is a parameter that controls the step size.

Functional gradient ascent is a more general approach. Instead of assuming a linear parameterization for Ψ, it just assumes that Ψ will be represented by a weighted sum of functions:

$$\Psi_m = \Psi_0 + \Delta_1 + \cdots + \Delta_m \ .$$

Each Δ_m is computed as a *functional gradient*:

$$\Delta_m = \eta_m \, E_{\mathbf{x},y} \left[\nabla_\Psi \log P(y \mid \mathbf{x}; \Psi)|_{\Psi_{m-1}} \right] \ .$$

The functional gradient indicates how we would like the function Ψ_{m-1} to change in order to increase the true log likelihood (i.e., on all possible points $(\mathbf{x},y)$). Unfortunately, we do not know the joint distribution $P(\mathbf{x},y)$, so we cannot evaluate the expectation $E_{\mathbf{x},y}[\cdot]$. We do have a set of training examples sampled from this joint distribution, so we can compute the value of the functional gradient at each of our training data points:

$$\Delta_m(y_i,\mathbf{x}_i) = \nabla_\Psi \sum_i \log P(y_i \mid \mathbf{x}_i;\Psi)\Big|_{\Psi_{m-1}} \quad .$$

We can then use these point-wise functional gradients to define a set of *functional gradient training examples*, $((\mathbf{x}_i,y_i),\Delta_m(y_i,\mathbf{x}_i))$, and then train a function $h_m(y,\mathbf{x})$ so that it approximates $\Delta_m(y_i,\mathbf{x}_i)$. Specifically, we can fit a regression tree h_m to minimize

$$\sum_i [h_m(y_i,\mathbf{x}_i) - \Delta_m(y_i,\mathbf{x}_i)]^2 \quad .$$

We can then take a step in the direction of this fitted function:

$$\Psi_m = \Psi_{m-1} + \eta h_m \quad .$$

Although the fitted function h_m is not exactly the same as the desired Δ_m, it will point in the same general direction (assuming there are enough training examples). So ascent in the direction of h_m will approximate true functional gradient ascent.

A key thing to note about this approach is that it replaces the difficult problem of maximizing the log likelihood of the data by the much simpler problem of minimizing squared error on a set of training examples. Friedman (2001) suggests growing h_m via a best-first version of the CART algorithm (Breiman et al., 1984; Friedman et al., 2000) and stopping when the regression tree reaches a pre-set number of leaves L. The pseudo-code of this algorithm is shown in Table 1. Overfitting is controlled by tuning L (e.g., by internal cross-validation).

In our experience, using L to control overfitting is a blunt tool that is hard to calibrate. In this paper, we instead introduce shrinkage into the algorithm for growing regression trees by adding a quadratic weight penalty. For each leaf in the regression tree h_m, the quantity that we minimize is the squared error of the examples $((\mathbf{x}_i,y_i),\Delta_m(y_i,\mathbf{x}_i))$ falling into this leaf plus a quadratic penalty:

$$\sum_i (\Delta_m(y_i,\mathbf{x}_i) - \hat{\delta})^2 + \lambda\hat{\delta}^2 \quad ,$$

where $\hat{\delta}$ is the output of this leaf and $\lambda > 0$ controls the strength of the penalty. Differentiating the above objective function with respect to $\hat{\delta}$ shows that the minimum is achieved at

$$\hat{\delta} = \frac{\sum_i \Delta_m(y_i,\mathbf{x}_i)}{\lambda + N} \quad , \tag{1}$$

where N is the total number of examples falling into this leaf. This has the nice interpretation that λ is an equivalent number of training examples with target values of 0. So this shrinks the leaf values (learned weights) toward zero. With this method, we can select a large number for L (the maximum number of leaves in the regression tree), and use λ to give fine control of overfitting. The algorithm shown in Table 1 can be adapted by using Equation 1 in the computation of function OUTPUT and function SQUAREDERROR. Experimental results show that this new algorithm works better and is more efficient than the original best-first version of the CART algorithm.

```
FITREGRESSIONTREE(Data, L)
// Data = {(x_i, y_i) : i = 1,...,N, x_i = (x_i1,...,x_ip)}
// NodeQueue is a priority queue of tree nodes where the first node has the minimum SplitScore
Root := FINDBESTSPLITATTRIBUTE(Data, NodeQueue)
NumLeaves := 1
while ((NumLeaves < L) AND NOTEMPTY(NodeQueue))
    Node := REMOVEFRONT(NodeQueue)
    TrueData := examples in Node whose values of SplitFeature are true
    FalseData := examples in Node whose values of SplitFeature are false
    TrueChild := FINDBESTSPLITATTRIBUTE(TrueData, NodeQueue)
    FalseChild := FINDBESTSPLITATTRIBUTE(FalseData, NodeQueue)
    SETCHILDNODES (Node, TrueChild, FalseChild)
    NumLeaves := NumLeaves + 1
end
return Root
end FITREGRESSIONTREE

FINDBESTSPLITATTRIBUTE(Data, NodeQueue)
SplitScore := 0, SplitFeature := 0
for j from 1 to p
    TrueData := {(x_i, y_i) ∈ Data : x_ij = 1}
    FalseData := {(x_i, y_i) ∈ Data : x_ij = 0}
    Gain := SQUAREDERROR(TrueData) + SQUAREDERROR(FalseData) − SQUAREDERROR(Data)
    if Gain < SplitScore
        SplitScore := Gain, SplitFeature := j
    end
end
Node := MAKELEAF(OUTPUT(Data), Data, SplitFeature, SplitScore)
if SplitFeature ≥ 1
    INSERT(Node, NodeQueue)
end
return Node
end FINDBESTSPLITATTRIBUTE
```

Table 1: Best-first version of the CART algorithm.

4. Training CRFs with Gradient Tree Boosting

In principle, it is straightforward to apply functional gradient ascent to train CRFs. All we need to do is to represent and train $\Psi(y_t, X)$ and $\Psi(y_{t-1}, y_t, X)$ as weighted sums of regression trees. Let

$$F^{y_t}(y_{t-1}, X) = \Psi(y_t, X) + \Psi(y_{t-1}, y_t, X)$$

be a function that computes the "desirability" of label y_t given values for label y_{t-1} and the input features X. There are K such functions F^k, one for each class label k. With this definition, the CRF has the form

$$P(Y|X) = \frac{1}{Z(X)} \exp \sum_t F^{y_t}(y_{t-1}, X) \ .$$

We now compute the functional gradient of $\log P(Y|X)$ with respect to $F^{y_t}(y_{t-1}, X)$. To simplify the computation, we replace X by $w_t(X)$, which is a window into the sequence X centered at $\mathbf{x}_t$. We will further assume, without loss of generality, that each window is unique, so there is only one occurrence of $w_t(X)$ in each sequence X.

Proposition 1 *The functional gradient of* $\log P(Y|X)$ *with respect to* $F^v(u, w_d(X))$ *is*

$$\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} = I(y_{d-1} = u, y_d = v) - P(y_{d-1} = u, y_d = v \mid w_d(X)) \ ,$$

where $I(y_{d-1} = u, y_d = v)$ *is 1 if the transition* $u \to v$ *is observed from position* $d-1$ *to position* d *in the sequence* Y *and 0 otherwise, and where* $P(y_{d-1} = u, y_d = v \mid w_d(X))$ *is the predicted probability of this transition according to the current potential functions.*

To demonstrate this proposition, we must first introduce the forward-backward algorithm for computing the normalizing constant $Z(X)$. We will assume that y_t takes the value $\perp$ for $t < 1$. Define the forward recursion by

$$\begin{aligned} \alpha(k,1) &= \exp F^k(\perp, w_1(X)) \\ \alpha(k,t) &= \sum_{k'} \exp F^k(k', w_t(X)) \cdot \alpha(k', t-1) \ , \end{aligned}$$

and the backward recursion by

$$\begin{aligned} \beta(k,T) &= 1 \\ \beta(k,t) &= \sum_{k'} \exp F^{k'}(k, w_{t+1}(X)) \cdot \beta(k', t+1) \ . \end{aligned}$$

The variables k and k' iterate over the possible class labels. The normalizer $Z(X)$ can be computed at any position t as

$$Z(X) = \sum_k \alpha(k,t)\beta(k,t) \ .$$

If we unroll the α recursion one step, we can also write this as

$$Z(X) = \sum_k \left[\sum_{k'} \alpha(k', t-1) \cdot \left[\exp F^k(k', w_t(X)) \right] \right] \beta(k,t) \ .$$

Table 2 shows the derivation of the functional gradient. In Equation 2, exactly one of the $F^{y_t}(y_{t-1}, w_t(X))$ terms will match $F^v(u, w_d(X))$, because $w_d(X)$ is unique. This term will have a derivative of 1, so we represent this by the indicator function $I(y_{d-1} = u, y_d = v)$. In Equation 3, we expand $Z(X)$ at position d using the forward-backward algorithm. Again because $w_d(X)$ is unique, only the product where $k' = u$ and $k = v$ will give a non-zero derivative, so this gives us Equation 4. The right-hand expression in Equation 4 is precisely the joint probability that $y_{d-1} = u$ and $y_d = v$ given X. **Q.E.D.**

If $w_d(X)$ occurs more than once in X, each match contributes separately to the functional gradient.

This functional gradient has a very satisfying interpretation: It is our error on a probability scale. If the transition $u \to v$ is observed in the training example, then the predicted probability $P(u, v \mid X)$

$$
\begin{aligned}
&\frac{\partial \log P(Y|X)}{\partial F^v(u, w_d(X))} \\
&= \frac{\partial}{\partial F^v(u, w_d(X))} \sum_t F^{y_t}(y_{t-1}, w_t(X)) - \log Z(X) \\
&= I(y_{d-1} = u, y_d = v) - \frac{\partial \log Z(X)}{\partial F^v(u, w_d(X))} \qquad (2) \\
&= I(y_{d-1} = u, y_d = v) - \frac{1}{Z(X)} \frac{\partial Z(X)}{\partial F^v(u, w_d(X))} \\
&= I(y_{d-1} = u, y_d = v) - \frac{1}{Z(X)} \frac{\partial}{\partial F^v(u, w_d(X))} \sum_k \left[\sum_{k'} \left[\exp F^k(k', w_d(X)) \right] \cdot \alpha(k', d-1) \right] \beta(k, d) \qquad (3) \\
&= I(y_{d-1} = u, y_d = v) - \frac{1}{Z(X)} \left[\exp F^v(u, w_d(X)) \right] \alpha(u, d-1) \beta(v, d) \qquad (4) \\
&= I(y_{d-1} = u, y_d = v) - P(y_{d-1} = u, y_d = v \mid X)
\end{aligned}
$$

Table 2: Derivation of the functional gradient.

should be 1 in order to maximize the likelihood. If the transition is not observed, then the predicted probability should be 0. Functional gradient ascent simply involves fitting regression trees to these residuals.

The pseudo code for our gradient tree boosting algorithm is shown in Table 3. The potential function for each class k is initialized to zero. Then M iterations of boosting are executed. In each iteration, for each class k, a set $S(k)$ of functional gradient training examples is generated. Each example consists of a window $w_t(X_i)$ on the input sequence, a possible class label k' at time $t-1$, and the target Δ value. A regression tree having at most L leaves is fit to these training examples to produce the function $h_m(k)$. This function is then added to the previous potential function to produce the next function. In other words, we are setting the step size $\eta_m = 1$. We experimented with performing a line search at this point to optimize η_m, but this is very expensive. So we rely on the "self-correcting" property of tree boosting to correct any overshoot or undershoot on the next iteration.

The sets of generated examples $S(k)$ can become very large. For example, if we have 3 classes and 100 training sequences of length 200, then the number of training examples for each class k is $3 \times 100 \times 200 = 60,000$. Although regression tree algorithms are very fast, they still must consider all of the training examples! Friedman (2001) suggests two tricks for speeding up the computation: sampling and influence trimming. In sampling, a random sample of the training data is used for training. In influence trimming, data points with Δ values close to zero are ignored. We did not apply either of these techniques in our experiments.

The most related work to ours is the virtual evidence boosting (VEB) algorithm developed by Liao et al. (2007) for training CRFs. Both VEB and our approach use boosting for feature induction. However, VEB is a "soft" version of maximum pseudo-likelihood training, where the observed values of neighborhood labels are not used, but the probability distribution over neighborhood labels is used as virtual evidence. Our approach is a true maximum log likelihood method that does not depend on the pseudo-likelihood approximation. Another difference is that VEB only uses decision stumps to induce simple features, while our approach uses regression trees to induce more complex feature combinations.

TREEBOOST($Data, L$)
// $Data = \{(X_i, Y_i) : i = 1, \ldots, N\}$
for each class k, initialize $F_0^k(\cdot,\cdot) = 0$
for $m = 1, \ldots, M$
  for class k from 1 to K
    $S(k) :=$ GENERATEEXAMPLES($k, Data, Pot_{m-1}$)
      // where $Pot_{m-1} = \{F_{m-1}^u : u = 1, \ldots K\}$)
    $h_m(k) :=$ FITREGRESSIONTREE($S(k), L$)
    $F_m^k := F_{m-1}^k + h_m(k)$
  end
end
return F_M^k for all k
end TREEBOOST

GENERATEEXAMPLES($k, Data, Pot_m$)
$S := \{\}$
for example i from 1 to N
  execute the forward-backward algorithm on (X_i, Y_i)
    to get $\alpha(k,t)$ and $\beta(k,t)$ for all k and t
  for t from 1 to T_i
    for k' from 1 to K
      $P(y_{i,t-1} = k', y_{i,t} = k \mid X_i) :=$
        $\dfrac{\alpha(k', t-1)\exp[F_m^k(k', w_t(X_i))]\beta(k,t)}{Z(X_i)}$
      $\Delta(k, k', i, t) := I(y_{i,t-1} = k', y_{i,t} = k) -$
        $P(y_{i,t-1} = k', y_{i,t} = k \mid X_i)$
      insert $((w_t(X_i), k'), \Delta(k, k', i, t))$ into S
    end
  end
end
return S
end GENERATEEXAMPLES

Table 3: Gradient tree boosting algorithm for CRFs.

5. Inference in CRFs

Once a CRF model has been trained, there are (at least) two possible ways to define a classifier $Y = H(X)$ for making predictions. First, we can predict the *entire sequence* Y that has the highest probability:

$$H(X) = \operatorname*{argmax}_Y P(Y|X) \ .$$

This makes sense in applications, such as part-of-speech tagging, where the goal is to make a coherent sequential prediction. This can be computed by the Viterbi algorithm (Rabiner, 1989), which has the advantage that it does not need to compute the normalizer $Z(X)$.

The second way to make predictions is to individually predict each y_t according to

$$H_t(X) = \operatorname*{argmax}_v P(y_t = v|X) \ ,$$

and then concatenate these individual predictions to obtain $H(X)$. This makes sense in applications where the goal is to maximize the number of individual y_t's correctly predicted, even if the resulting predicted sequence Y is incoherent. For example, a predicted sequence of parts of speech might not be grammatically legal, and yet it might maximize the number of individual words correctly classified. $P(y_t|X)$ can be computed by executing the forward-backward algorithm as

$$P(y_t|X) = \frac{\alpha(y_t,t)\beta(y_t,t)}{Z(X)} \ .$$

6. Handling Missing Values in CRFs with Gradient Tree Boosting

In some problem settings (e.g., activity recognition, sensor networks), the problem of missing values in the inputs can arise. The values of input features can be missing for a wide variety of reasons. Sensors may break or the sensor data feed may be lost or corrupted. Alternatively, input observations may not have been measured in all cases because, for example, they are expensive to obtain. Many methods for handling missing values have been developed for standard supervised learning, but many of them have not been tested on SSL problems. Recently, Sutton et al. (2006) used feature bagging method to deal with SSL problems where highly indicative features may be missing in the test data. A single CRF trained on all the features will be less robust, because the weights of weaker features will be undertrained. Feature bagging method divides all the original features into a collection of complementary and possibly overlapped feature subsets. Separate CRFs are trained on each subset and then combined.

With gradient tree boosting, a CRF is represented as a forest of regression trees. There exist very good methods for handing missing values when growing regression trees, which include instance weighting method of C4.5 (Quinlan, 1993) and surrogate splitting of CART (Breiman et al., 1984). An advantage of training CRFs with gradient tree boosting is that these missing values methods can be used directly in the process of generating regression trees over the functional gradient training examples.

6.1 Instance Weighting

The instance weighting method (Quinlan, 1993), also known as "proportional distribution", assigns a weight to each training example, and all splitting decisions are based on weighted statistics. Initially, each example has a weight of 1.0. When selecting a feature to split on, each boolean feature x_j is evaluated based on the expected weighted squared error of the split using only the training examples for which x_j is not missing. The best feature x_{j*} is chosen, and the training examples for which x_{j*} is not missing are sent to the appropriate child node. Suppose that n_{left} examples are sent to the left child and n_{right} examples are sent to the right child. The remaining training examples (i.e., those for which x_{j*} is missing) are sent to *both* children, but with reduced weight. The weight

of each example sent to the left child is multiplied by $n_{left}/(n_{left}+n_{right})$. Similarly, the weight of each example sent to the right child is multiplied by $n_{right}/(n_{left}+n_{right})$.

At test time, when the test example reaches the test on feature x_{j*}, if the feature value is present, then the example is routed left or right in the usual way. But if x_{j*} is missing, then the example is sent to both children (recursively). Let $\hat{y}_{left}$ be the predicted value computed by the left subtree and $\hat{y}_{right}$ be the predicted value computed by the right subtree. Then the value predicted by node $j*$ is the weighted average of these predictions:

$$\hat{y} = \frac{n_{left}\hat{y}_{left} + n_{right}\hat{y}_{right}}{n_{left}+n_{right}} \quad .$$

Instance weighting assumes that the training and test examples missing x_{j*} will on average behave exactly like the training examples for which x_{j*} is not missing.

6.2 Surrogate Splitting

The surrogate splitting method (Breiman et al., 1984) involves separate procedures during training and testing. During training, as the regression tree is being constructed (in the usual top-down, greedy way), the key step in the learning algorithm is to choose which feature to split on. Each boolean feature x_j is evaluated based only on the training examples that have non-missing values for that feature, and the best feature, x_{j*} is chosen. Each of the remaining features $j' \neq j*$ is then evaluated to determine how accurately it can predict the value of x_{j*}, and the features are sorted according to their predictive power. This sorted list of features, called the surrogate splits, is stored in the node.

At test time, when test example x is processed through the regression tree, if x_{j*} is not missing, then the example is processed as usual by sending it to the left child if x_{j*} is false and to the right child if x_{j*} is true. However if x_{j*} is missing, then surrogate split features are examined in order until a feature j' is found that is not missing. The value of this feature determines whether to branch left or right.

7. Experimental Results

We implemented gradient tree boosting algorithm for CRFs and compared it to McCallum's Mallet system (McCallum, 2002) on several data sets. We call our algorithm TREECRF. We use TREECRF-FB for the TREECRF with forward-backward predictions and TREECRF-V for the TREECRF with Viterbi predictions. MALLET denotes the Mallet package with McCallum's feature induction algorithm (McCallum, 2003) turned on. Similarly, we use MALLET-FB and MALLET-V for the MALLET with forward-backward predictions and Viterbi predictions respectively. We also used the Mallet package to train standard CRFs without feature induction. We call it BASELINE, which serves as the baseline method. As before, BASELINE-FB denotes BASELINE with forward-backward predictions and BASELINE-V denotes BASELINE with Viterbi predictions. Note that MALLET-FB algorithm and BASELINE-FB algorithm are not implemented in the original Mallet package. Instead we implemented them ourselves.

TREECRF, MALLET and BASELINE have parameters that must be set by the user. For all these algorithms, the user must set (a) the window size, (b) the order of the Markov model, which is set to be 1 in our experiments, and (c) the number of iterations to train. For TREECRF, the only

additional parameter is either the maximum number of leaves L in the regression trees using the best-first version of CART, or the regularization constant λ for the shrinkage alternative. For MALLET, the parameters are (a) the regularization penalty for squared weights (called the variance), (b) the number of iterations between feature inductions (kept constant at 8), (c) the number of features to add per feature induction (kept constant at 500), (d) the true label probability threshold (kept constant at 0.95), (e) the training proportions (kept constant at 0.2, 0.5, and 0.8). For BASELINE, the only additional parameter is the variance as in MALLET. Except for the variance, we kept all of MALLET's parameters fixed at the values recommended by Andrew McCallum (personal communication). We did not optimize the window size, but instead employed values that have been used in previous studies. The chosen sizes are given in the following section. To set the remaining parameters, we manually tried the following settings and chose the setting that gave the best internal cross-validation performance:

- Number of leaves in regression trees: 30, 50, 75, 100,
- TreeCRF regularization constant: 0, 5, 10, 20, 40, 80,
- Weight variance prior in Mallet package: 1, 5, 10, 20.

Throughout the experiments, we measured the performance by computing the prediction accuracy of individual labels, rather than individual sequences. McNemar's test is employed to assess the statistical significance of these results.

7.1 Data Sets

Protein Secondary Structure Benchmark (Qian and Sejnowski, 1988). Each observation sequence is a string of amino acid residues, and the corresponding output sequence is a string over the 3-letter alphabet $\{\alpha, \beta, \gamma\}$, where α indicates alpha helix, β indicates a beta sheet or beta turn, and γ indicates all other secondary structure types. There are 20 possible amino acid residues, and we represent each residue by a set of 20 indicator variables. There is a training set of 111 sequences and a test set of 17 sequences. An 11-residue sliding window is used in our experiments.

NETtalk Data Set. The original NETtalk task (Sejnowski and Rosenberg, 1987) is to assign a combination of phoneme and stress to each letter of the word so that the word is pronounced correctly. However, there are 140 legal phone-stress combinations, which gives a very large label space. Neither TREECRF nor MALLET is sufficient enough to work with such a large label space. Hence, we chose to study only the problem of assigning one of five possible stress labels to each letter. The labels are '2' (strong stress), '1' (medium stress), '0' (light stress), '$<$' (unstressed consonant, center of syllable to the left), and '$>$' (unstressed consonant, center of syllable to the right).

Each input sequence is an English word, a string of letters over the 26 letter alphabet. Each input observation is represented by 26 boolean indicator variables. There are 1000 training words and 1000 test words in our standard training and test sets. We employed a window size of 13 (window width of 6).

Hyphenation Data Set. The hyphenation task is to insert hyphens into words at points where it is legal to break a word for a new line. This problem appears widely in many word processing programs. The input sequences are English words, encoded as for the NETtalk task. The output class label has only two values to indicate whether or not a hyphen may legally follow the current

	TREECRF-FB		TREECRF-V	
	Shrinkage	Original	Shrinkage	Original
Protein	64.52**	62.70	62.05***	59.20
NETtalk	85.18***	84.08	85.20***	84.18
Hyphen	92.20	92.20	91.76	92.07
FAQ ai-general	95.65	95.69	95.72	96.02***
FAQ ai-neural	99.02	98.97	99.20***	99.05
FAQ aix	94.00	94.02	95.26*	95.15

Table 4: Performance comparison of TREECRF with different regression tree fitting algorithms. Entries marked with one or more stars are statistically significantly better than the alternative method. Specifically, * means $p < 0.025$, ** means $p < 0.005$ and *** means $p < 0.001$ according to McNemar's test.

letter. We manually constructed a training set of 1951 words and a test set of 908 words. The input window size is set to be 6 (i.e., 3 letters on either side of the potential hyphen location).

Usenet FAQs Data Sets. Each of the FAQ data sets consists of Frequently Asked Questions files for a Usenet newsgroup (McCallum et al., 2000). The FAQs for each newsgroup are divided in separate files: ai-general has 7 files, ai-neural has 7 files, and aix has 5 files. Every line of an FAQ file is labeled as either part of the header, a question, an answer, or part of the tail. Hence, each $\mathbf{x}_t$ consists of a line in the FAQ file, and the corresponding $y_t \in \{\text{header, question, answer, tail}\}$. The measure of accuracy is the number of individual lines correctly classified. McCallum provided us with the definitions of 20 features for each line $\mathbf{x}_t$. We made a slight correction to one of the features, so our results are not directly comparable to his. The size of the sliding window used here is 1. For each newsgroup, performance was measured by leave-1-out cross-validation: the CRF was trained on all-but-one of the files and tested on the remaining file. This was repeated with each file, and the results averaged.

7.2 Performance of Shrinkage in Regression Tree Generation

To evaluate the effectiveness of shrinkage in the regression tree fitting algorithm, we fixed L, the maximum number of leaves in regression trees, to be 100, and applied internal cross-validation to choose the best regularization constant λ. For purposes of comparison, we also implemented the original best-first regression tree generation algorithm. Internal cross-validation was employed to select the best value for L.

We ran these two implementations of TREECRF on each data set. The best performance of both forward-backward predictions and Viterbi predictions are reported as percentages, as shown in Table 4. There are 12 pairs of comparisons (6 data sets with 2 prediction algorithms). In six of them, TREECRF with shrinkage does statistically better than TreeCRF without shrinkage. In five of them, the performance of these two versions of TREECRF is statistically indistinguishable. In only one of them, TREECRF without shrinkage does statistically better than TREECRF with shrinkage. Based on the results of these experiments, we decided to only employ TREECRF with shrinkage in the remaining experiments.

		Protein	NETtalk	Hyphen	FAQ ai-general	FAQ ai-neural	FAQ aix
Accuracy (%)	TREECRF	64.52*	85.20**	92.20**	**95.65** **	99.20**	95.26**
	MALLET	64.43*	**85.94****	92.10**	92.70	99.31*	95.28**
	BASELINE	62.44	82.81	88.86	92.70	99.41	94.04
Cumulative CPU Seconds	TREECRF	419.6	454.6	39.2	3921.9	2177.7	2636.1
	MALLET	786.9	941.4	66.4	484.1	237.2	125.5
	BASELINE	32.8	13.7	8.8	63.0	40.3	34.1
Iterations	TREECRF	142	169	58	214	84	158
	MALLET	123	167	69	188	181	150
	BASELINE	66	34	47	128–195	72–112	80–140

Table 5: Performance of TREECRF, MALLET, and BASELINE on each data set. Entries marked with one or more stars are statistically significant than BASELINE. Specifically, * means $p < 0.005$, ** means $p < 0.001$ according to McNemar's test. Bolded numbers indicate the statistically better prediction accuracy between TREECRF and MALLET. The BASELINE method stops training if the optimization of loss functions converges. So for each FAQ data set, different training set may have different number of training iterations. Here we gave out the range of number of training iterations for each FAQ data set.

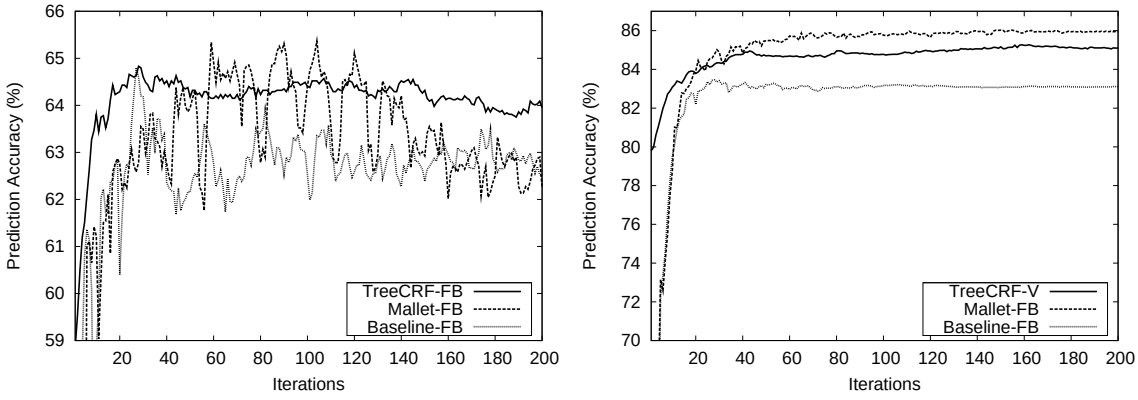
7.3 Comparison between TREECRF and MALLET

TREECRF and MALLET are the two leading CRF training methods that have feature induction capability. Here we compare the prediction accuracy and training speed of these two methods on each available data set. We also compare TREECRF and MALLET with the BASELINE method. For each method, internal cross-validation is applied to select the parameters that give the best performance of both forward-backward predictions and Viterbi predictions. The results reported here for each method are based on the prediction algorithm that gives higher prediction accuracy. All experiments were run on machines with 2.4 GHz Intel Xeon processors, 512KB cache, and 4GB memory.

Prediction Accuracy. Table 5 summarizes the prediction accuracy of TREECRF, MALLET, and BASELINE on each data set. McNemar's tests show that on four of the data sets, that is, protein, hyphen, FAQ ai-neural and FAQ aix, the difference between the prediction accuracy of TREECRF and MALLET is not statistically significant. On the FAQ ai-general data set, the prediction accuracy of TREECRF is statistically better than that of MALLET($p < 0.001$). Only on the NETtalk data set is the prediction accuracy of MALLET statistically better than that of TREECRF ($p < 0.05$). In comparison with the baseline method, the prediction accuracy of TREECRF and MALLET is statistically better than that of BASELINE in most cases. On the FAQ ai-general data set, the difference between MALLET and BASELINE is not statistically significant. Only on the FAQ ai-neural data set is the prediction accuracy of BASELINE statistically better than that of both TREECRF and MALLET.

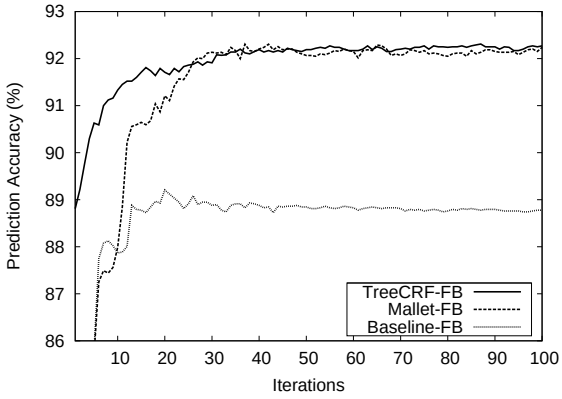
Figure 1 plots the prediction accuracy of TREECRF, MALLET and BASELINE as a function of the number of training iterations. One worrying aspect of MALLET is that the performance curve exhibits a high degree of fluctuation, which is clearly shown on Figure 1a, 1d, 1e and 1f. This is presumably due to the effect of introducing new features. But it also suggests that it will be difficult to find the optimal stopping points for avoiding overfitting.

Training Speed. It is difficult to directly compare the CPU time of these two methods, because TREECRF is written in C++ while MALLET is written in Java. However, comparing the CPU time

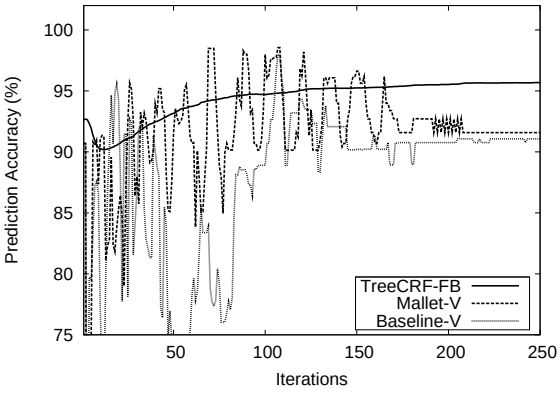


(a) Protein

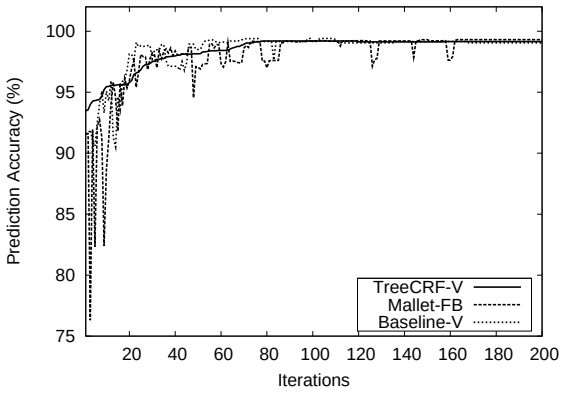
(b) NETtalk



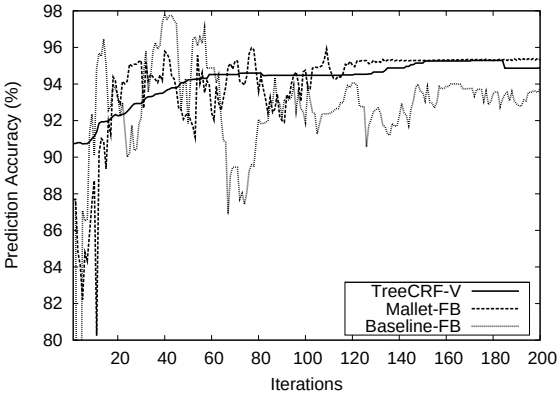
(c) Hyphen



(d) FAQ ai-general

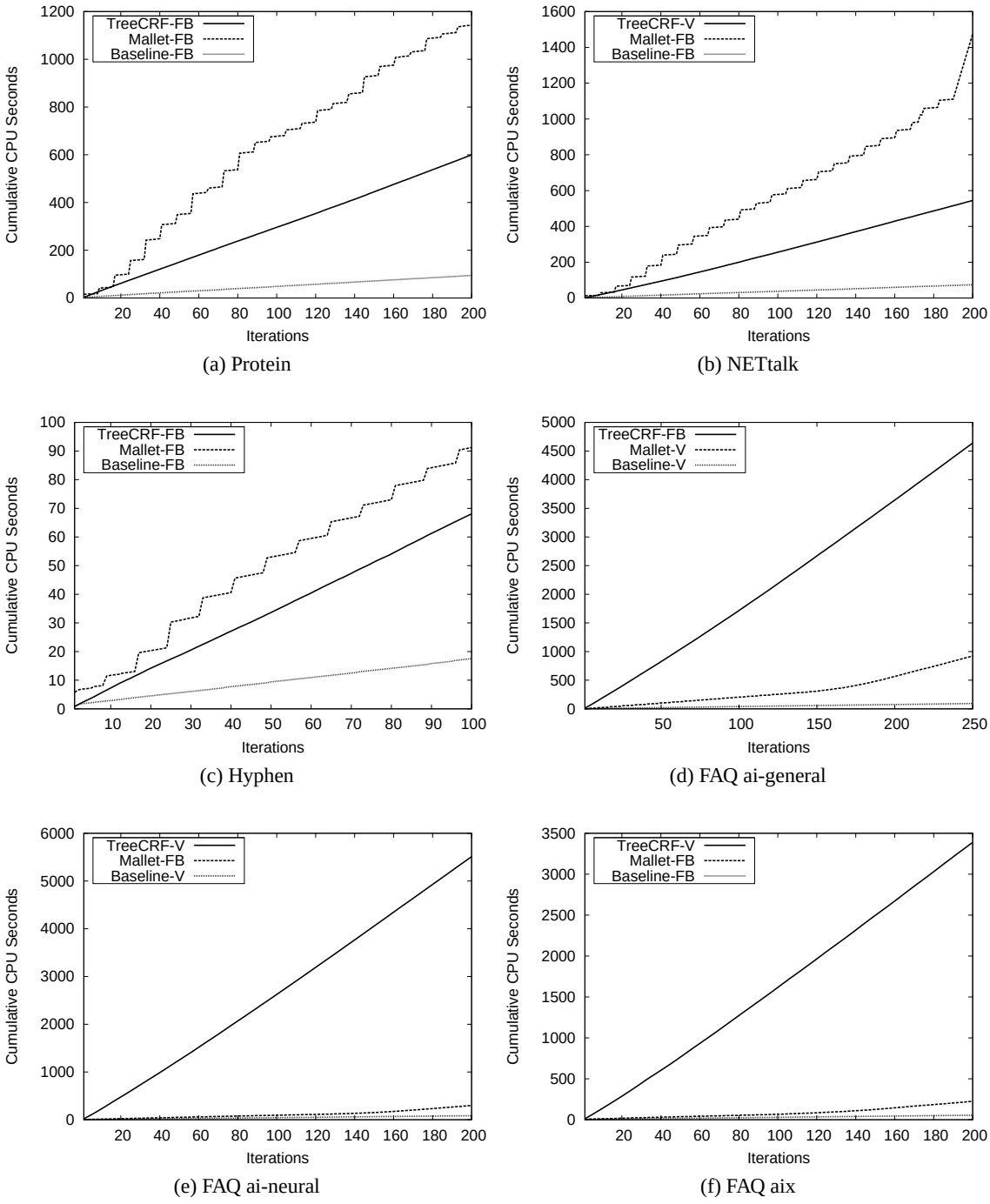


(e) FAQ ai-neural



(f) FAQ aix

Figure 1: Comparison of prediction accuracy on each data set.



(a) Protein (b) NETtalk

(c) Hyphen (d) FAQ ai-general

(e) FAQ ai-neural (f) FAQ aix

Figure 2: Comparison of cumulative CPU time on each data set.

Data Set	Average Length	Number of Features	Forward-Backward Seconds		Feature Induction Seconds	
			TREECRF	MALLET	TREECRF	MALLET
Protein	163	231	1.493	0.736	1.433	48.889
NETtalk	7	351	0.622	0.589	2.049	25.983
Hyphen	6	162	0.324	0.307	0.332	4.621
FAQ ai-general	1580	20	18.927	0.780	1.562	1.211
FAQ ai-neural	1832	20	26.998	0.526	1.894	1.656
FAQ aix	1806	20	16.658	0.352	1.199	1.123

Table 6: Comparison of average CPU seconds spent per iteration on forward-backward algorithm and feature induction algorithm in TREECRF and MALLET for each data set.

on different data sets can still give us some insight into the properties of these two methods. Figure 2 shows the number of cumulative CPU seconds consumed by these two methods on each data set. First, we can see that TREECRF scales linearly in the number of training iterations, because the cumulative CPU time has a constant slope. This makes sense, because for each potential function, only one regression tree is generated in each training iteration. Regression tree evaluations from previous iterations are cached so that they do not need to be re-evaluated. Without caching, the cumulative CPU curves for TREECRF would rise quadratically. Second, as shown in Figure 2a, 2b and 2c, TREECRF runs faster than MALLET on protein, NETtalk and hyphen data sets. But it is much slower than MALLET on FAQ data sets as shown in Figure 2d, 2e and 2f. The actual time required for each method to reach its peak performance on each data set is given in Table 5. Again we see that on the protein, NETtalk, and hyphen data sets, the time required for MALLET to reach its peak performance is about twice that of TREECRF. However, on the FAQ data sets, the time required for TREECRF to reach its peak performance is about 10-20 times more than for MALLET. BASELINE is faster than both TREECRF and MALLET as shown in Figure 2 and Table 5.

Analysis and Discussion. We can explain the training speed difference between TREECRF and MALLET by examining the details of these two methods. In both of them, most of the CPU time is spent on two major computations: forward-backward inference and feature induction/tree growing. The relative proportion of these two computations varies from problem to problem. To measure this, we instrumented both TREECRF and MALLET to track the amount of CPU time spent on each of these two computations. Table 6 shows that on domains with short sequences (Protein, NETtalk, and Hyphen), the time spent by both algorithms on forward-backward inference is about the same. But for domains with very long sequences, TREECRF consumes much more CPU time in forward-backward inference. Conversely, in domains with a small number of basic features (the FAQ data sets), the two methods consume roughly the same amount of CPU time in feature induction. But in domains with a large number of basic features, TREECRF is much more efficient than MALLET.

Why would the forward-backward cost of TREECRF be larger than for MALLET? TREECRF and MALLET use almost the same implementation of forward-backward algorithm except that in TREECRF the values of the potential functions at each position of the sequences are computed by evaluating the gradient regression trees generated in the current training iteration, while in MALLET those values are obtained by computing dot products of vectors, which is faster than tree evaluation. We hypothesize that the regression trees are more expensive to evaluate, not only because dot prod-

ucts are easier to compute than tree evaluations, but also possibly because of the reduced memory locality of regression trees.

Why would feature induction be more expensive in MALLET? In each feature induction iteration, MALLET considers conjoining all of the basic features to each of the existing compound features. Hence, if there are n basic features and C compound features, this costs nC. Furthermore, C grows over time, so the cost of feature induction gradually increases. In the cumulative CPU time plots of Figure 2, the "steps' in the "staircase" correspond to the feature induction iterations. In TREECRF, the cost of feature induction is the cost of growing a regression tree, which depends on the number of basic features n and the number of internal nodes in the tree L. This cost is nL, which remains constant across the iterations.

To verify our conjectures about the computational complexity of TREECRF and MALLET, we generated synthetic training data sets using a hidden Markov model (HMM) with 3 labels $\{l_1, l_2, l_3\}$ and 24 possible observations $\{o_1, \ldots, o_{24}\}$. To specify the observation distribution, for each label l_i, we randomly draw an observation from the set $\{o_{i*8-7}, \ldots, o_{i*8}\}$ with probability 0.6 and randomly draw an observation from the complement of this set with probability 0.4. The transition distribution is defined as $P(y_t = l_i \mid y_{t-1} = l_i) = 0.6$ and $P(y_t = l_j \mid y_{t-1} = l_i) = 0.2$ if $i \neq j$.

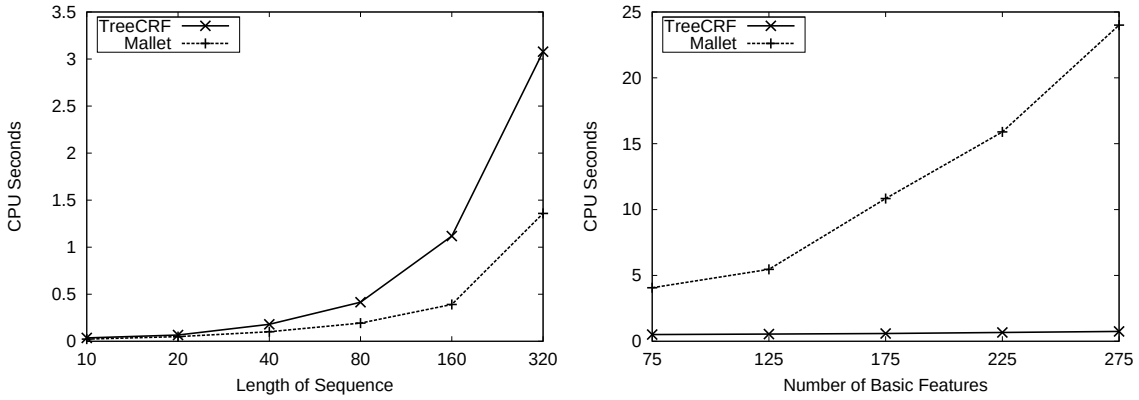
In order to measure the complexity of the forward-backward algorithm, we tried sequence lengths of 10, 20, 40, 80, 160 and 320. For each sequence length, we generated a training data set with 100 sequences and employed a sliding window of size 3. TREECRF and MALLET are run on each of these training data sets. Figure 3a shows the average CPU seconds spent per iteration on the forward-backward algorithm by these two methods. We see that the forward-backward algorithm in TREECRF implementation scales faster than that in MALLET implementation as the length of sequence increases.

In order to measure the complexity of the feature induction algorithms, we generated a training data set with 100 sequences. The length of each sequences is 100. We tried sliding window sizes of 3, 5, 7, 9 and 11, so that the number of input features at each sequence position takes the values of 75, 125, 175, 225 and 275 (because each input observation is represented by 25 boolean indicator variables). TREECRF and MALLET are run for each sliding window size. Figure 3b shows the average CPU seconds spent per iteration on the feature induction algorithm by these two methods. It is clear that the feature induction algorithm in MALLET spends more and more CPU time than that in TREECRF as the number of basic features increases. In all the experiments on synthetic data sets, TREECRF uses regression trees of maximum 100 leaves and shrinkage constant 40. MALLET uses weight variance prior 20.

This analysis suggests that the performance of TREECRF could be improved by "flattening" the ensemble of regression trees to compute the corresponding vector of features and vector of weights. Then the cost of potential function evaluations would be similar to that of MALLET, and we would have a method that was faster than both the current TREECRF and MALLET implementations.

7.4 Experimental Studies of Missing Values in TREECRF

We performed a series of experiments to evaluate the effectiveness of methods for handling missing values in TREECRF algorithm. In addition to the instance weighting and surrogate splitting methods described above, we also studied two simpler methods: imputation and indicator features. Let $x_{tj}, j = 1, \ldots, n$ be the input features describing a particular input observation $\mathbf{x}_t$. Imputation and indicator features are defined as follows:



(a) Forward-backward algorithm (b) Feature induction algorithm

Figure 3: Comparison of average CPU seconds spent per iteration on forward-backward algorithms and feature induction algorithms by TREECRF and MALLET.

Imputation: when a feature value x_{tj} is missing, it is replaced with the most common value for x_j in the training data among those feature values that are not missing. This strategy can be viewed as substituting the most likely value of x_j a priori or alternatively as substituting the value of x_j least likely to be informative.

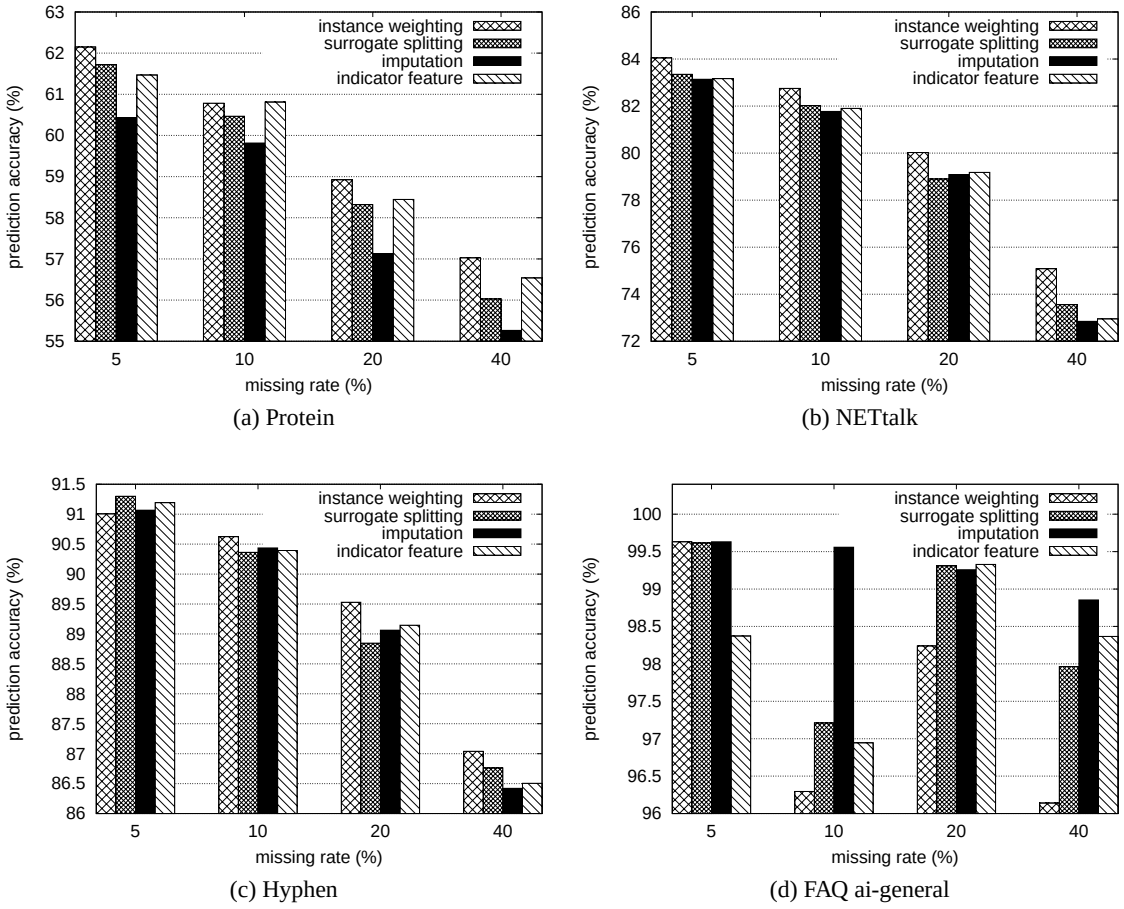
Indicator Features: a boolean feature $\tilde{x}_{tj}$ is introduced for each feature x_{tj} such that if x_{tj} is present, $\tilde{x}_{tj}$ is false. But if x_{tj} is missing, then $\tilde{x}_{tj}$ is true and x_{tj} is set to a fixed chosen value, typically 0. Indicator features make sense when the fact that a value is missing is itself informative. For example, if x_{tj} represents a temperature reading, it may be that extremely cold temperature values tend to be missing because of sensor failure.

We adopted a first-order Markov model in all the following experiments and employed an internal hold-out method to set the other parameters: Two-thirds of the original training set was used as sub-training set and the other one third was used as development set to choose parameter values. Final training was performed using the entire training set.

For each learning problem, we took the chosen training and test sets and inject missing values at rates of 5%, 10%, 20% and 40%. For a given missing rate, we generate five versions of the training set and five versions of the test set. A CRF is then trained on each of the training sets and evaluated on each of the test sets (for a total of 5 CRFs and 25 evaluations per missing rate). The label sequences are predicted by the forward-backward algorithm (i.e., we compute $\hat{y}_t = \text{argmax}_{y_t} P(y_t|X)$ for each t separately). Prediction accuracy is based on the number of individual labels correctly predicted in the label sequences. The final prediction accuracy is the average of all 25 cases.

To test the statistical significance of the differences among the four methods, we performed an analysis of deviance based on the generalized linear model discussed by Agresti (1996). We fit a logistic regression model

$$\log \frac{P(y_t = \hat{y}_t)}{1 - P(y_t = \hat{y}_t)} = \delta_1 m_1 + \delta_2 m_2 + \delta_3 m_3 + \sum_\ell \sigma_\ell S_\ell \ ,$$



(a) Protein (b) NETtalk

(c) Hyphen (d) FAQ ai-general

Figure 4: Performance of missing values methods for different missing rates.

where m_1, m_2, and m_3 are boolean indicator variables that specify which missing values method we are using and the S_ℓ's are indicator variables that specify which of the five training sets we are using. If $m_1 = m_2 = m_3 = 0$, then we are using instance weighting, which serves as our baseline method. If $m_1 = 1$, this indicates surrogate splitting, $m_2 = 1$ indicates imputation, and $m_3 = 1$ indicates the indicator feature method. Consequently, the fitted coefficients δ_1, δ_2, and δ_3 indicate the change in log odds (relative to the baseline) resulting from using each of these missing values methods. We can then test the hypothesis $\delta_i \neq 0$ against the null hypothesis $\delta_i = 0$ to determine whether missing values method i is different from the baseline method.

This statistical approach controls for variability due to the choice of the training set (through the σ_ℓ's) and variability due to the size of the test set.

Protein Secondary Structure Prediction. Figure 4a shows that instance weighting achieves the best prediction accuracy for each of the different missing rates. Table 7a shows that the base line missing values method, instance weighting, is statistically better than the other three missing values methods in most cases. In other cases, it is as good as other methods.

Missing rate	Surrogate splitting	Imputation	Indicator feature
5%	−0.018	−0.072*	−0.028*
10%	−0.013	−0.040*	0.001
20%	−0.025*	−0.074*	−0.020*
40%	−0.041*	−0.072*	−0.020*

(a) Protein

Missing rate	Surrogate splitting	Imputation	Indicator feature
5%	−0.051*	−0.066*	−0.064*
10%	−0.051*	−0.067*	−0.059*
20%	−0.069*	−0.057*	−0.052*
40%	−0.080*	−0.116*	−0.111*

(b) NETtalk

Missing rate	Surrogate splitting	Imputation	Indicator feature
5%	0.036*	0.007	0.023
10%	−0.031*	−0.022	−0.027*
20%	−0.071*	−0.049*	−0.040*
40%	−0.024*	−0.054*	−0.047*

(c) Hyphen

Missing rate	Instance weighting	Surrogate splitting	Indicator feature
5%	−8.824E−16	−0.043	−1.499*
10%	−2.161*	−1.867*	−1.961*
20%	−0.874*	0.072	0.100
40%	−1.243*	−0.584*	−0.359*

(d) FAQ ai-general

Table 7: Estimation of the coefficients corresponding to different missing values methods and statistical test results. In FAQ ai-general problem, imputation was the baseline method, so the coefficient values give the log odds of the change in accuracy relative to imputation. * means that the parameter value is statistically significantly different from zero ($p < 0.05$).

NETtalk Stress Prediction. In Figure 4b, we see that instance weighting does better than the other three missing values methods for all the different missing rates. The statistical tests reported in Table 7b show that the baseline method, instance weighting, is statistically better than each of the other missing value methods in all cases.

Hyphenation. Figure 4c shows that instance weighting is the best missing values method except for a missing rate of 5%. Statistical tests shown in Table 7c tell us that for missing rate of 5%, surrogate splitting is the best missing values method and the other three methods are not statistically significantly different from each other. For a missing rate of 10%, instance weighting and imputation are statistically better than the other two methods (and indistinguishable from each other). For missing rates of 20% and 40%, instance weighting is statistically better than the other three methods.

FAQ Document Segmentation. This task is based on the ai-general Usenet FAQ data set as we discussed before. We treat the first 6 files as the training set and the seventh file as the test set. The input window contains only the features corresponding to a single line in the file (window half-width of 0). Unlike in the previous data sets, instance weighting is no longer the best missing values method, as shown in Figure 4d. Instead, imputation performs very well for various missing value rates. Table 7d shows that imputation is statistically the best missing values method. For missing rates of 10% and 40%, it is statistically better than the other three methods. For a missing rate of 5%, it does as well as instance weighting and surrogate splitting. For a missing rate of 20%, it does as well as surrogate splitting and indicator features.

Analysis and Discussion. The four missing values methods are based on different assumptions about the input data. Imputation assumes that the most frequent value of a feature is the least informative and therefore presents the lowest risk of introducing errors into the learning process. Missing values are injected prior to converting the input features to binary. Hence, in the protein data set, missing values are introduced by choosing an amino acid residue position in the observa-

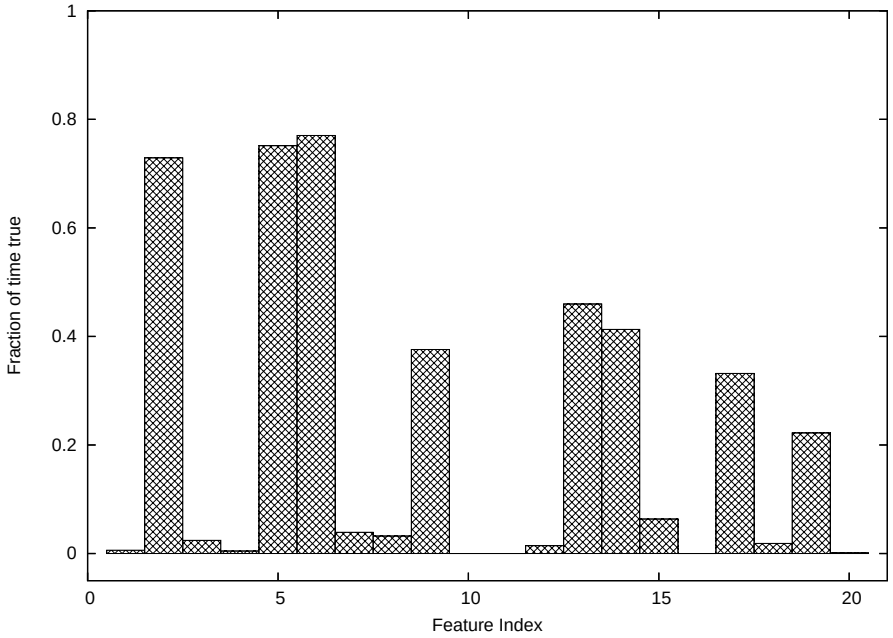


Figure 5: Fraction of the time that each FAQ feature is true (versus false). Features 1, 3, 4, 7, 8, 10, 11, 12, 16, 18, and 20 are rarely true.

tion sequence and setting all 20 boolean indicator features that represent that position to missing. Similarly, in the NETtalk and hyphenation problems, a letter is made to be missing by setting all 26 indicator features for that letter to missing. Similarly, imputation is computed at the amino acid or letter level, not at the level of boolean features. However, in the Usenet FAQ data set, since the binary features are not exclusive, imputation is computed at the level of boolean features. In the case of protein sequences, imputation will replace missing values with the most frequently-occurring amino acid, which is alanine, code 'A'. Alanine tends to form alpha helices, so this may cause the learning algorithms to over-predict the helix class, which may explain why imputation performed worst on the protein data set. In the case of English words, the most common letter is 'E', and it does not carry much information either about pronunciation or about hyphenation, so this may explain why imputation worked well in the NETtalk and hyphenation problems. Finally, in the ai-general FAQ data set, most of the features exhibit a highly skewed distribution, so that one feature value is much more common than another, as shown in Figure 5. Hence, in most cases, imputation with the most common feature value will supply the correct missing value. This may be why it worked best on that data set.

The indicator feature approach is based on the assumption that the presence or absence of a feature is meaningful (e.g., in medicine, a feature could be missing because a physician explicitly chose not to measure it). Because features were marked as missing completely at random, this is not true, so the indicator feature carries no positive information about the class label. However, in cases where imputation causes problems, the indicator feature approach may help prevent those problems by being more neutral. The learning algorithm can learn that if the indicator feature is set, then the actual feature value should be ignored. This may explain why the indicator feature method works slightly better in most cases than the imputation method.

The surrogate splitting method assumes that the input features are correlated with one another, so that if one feature is missing, its value can be computed from another feature. The protein, NETtalk, and hyphenation data sets have a single input feature for each amino acid or letter. Hence, if this input feature is missing, then there is no information about that position in the sequence. The only exception to this would be if there were strong correlations between successive amino acids or letters. However, such strong correlations do not exist much either in protein sequences or in English, with the possible exception of the letter 'q', which is always followed by 'u'. Note that the converse is not true: 'u' is not always preceded by 'q'. Based on these considerations, we would not expect surrogate splitting to work well in these domains, and it does not.

In the FAQ data set, each line is described by 20 features computed from the words in that line. In the experiment, each of these 20 features could be independently marked as missing, which is a bit unrealistic, since presumably the real missing values would involve some loss or corruption of the words making up the line, and this would affect multiple features. The 20 features do have some redundancy, so we would expect that surrogate splitting should work well, and it does for 5% and 20% missing rates.

The instance weighting method assumes that the feature values are missing at random and that the other features provide no redundant information, so the most sensible thing to do is to marginalize away the uncertainty about the missing values. Our experiments show that this is a very good strategy in all cases except for the FAQ data set, where the features are somewhat redundant.

8. Conclusions

In this paper, we presented TREECRF, a novel method for training conditional random fields based on gradient tree boosting. TREECRF has the ability to construct very complex feature conjunctions from basic features and scales much better than methods based on iterative scaling and simple gradient descent. It appears to match the L-BFGS algorithm implemented in MALLET, which also gives dramatic speedups when there are many potential features. In our experiments, TREECRF is as accurate as MALLET on four data sets, more accurate on one data set and less accurate on one data set. Its feature induction method is faster than that of MALLET for problems with a large number of features. But its forward-backward implementation is slower than that of MALLET for really long sequences. In addition, TREECRF is easier to implement and tune. It introduces only one tunable parameter (either the maximum number of leaves permitted in each regression tree or the regularization constant), whereas MALLET has many more parameters to consider. It is easier for the TREECRF to find the optimal stopping point to avoid overfitting, since its performance improves smoothly, while that of MALLET fluctuates wildly. Combining the benefit of these two methods will be a promising direction to pursue.

TREECRF also provides us with extra ability to handle missing data with instance weighting and surrogate splitting methods, which are not available in MALLET and other CRF training algorithms. The experiments suggest that when the feature values are missing at random, the instance weighting approach works very well. In the one domain where instance weighting did not work well, imputation was the best method. The indicator feature method was also very robust. The method of surrogate splitting was the most expensive method to run and the least accurate. Hence, we do not recommend using surrogate splits with conditional random fields. The good performance of the indicator features and imputation methods is encouraging, because these methods can be applied with all known methods for sequential supervised learning, not only with gradient tree

boosting. Since there is no one best method for handling missing values, as with many other aspects of machine learning, preliminary experiments on subsets of the training data are required to select the most appropriate method.

Acknowledgments

The authors would like to thank the anonymous reviewers and the action editor for their constructive input. We also gratefully acknowledge the support of NSF grants IIS-0083292 and IIS-0307592. Some of the material in this paper was first published at ICML-2004 (Dietterich et al., 2004).

References

Alan Agresti. *An Introduction to Categorical Data Analysis*. Wiley, New York, 1996.

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines. In Tom Fawcett and Nina Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, pages 3–10. AAAI Press, 2003.

Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society B*, 36(2):192–236, 1974.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984.

Thomas G. Dietterich, Adam Ashenfelter, and Yaroslav Bulatov. Training conditional random fields via gradient tree boosting. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, pages 217–224, Banff, Canada, 2004. ACM Press.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*, pages 148–156. Morgan Kaufmann, 1996.

Jerome Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6): 721–741, Nov. 1984.

John M. Hammersley and Peter Clifford. Markov fields on finite graphs and lattices. Technical report, Unpublished, 1971.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML 2001)*, pages 282–289. Morgan Kaufmann, 2001.

Lin Liao, Tanzeem Choudhury, Dieter Fox, and Henry A. Kautz. Training conditional random fields using virtual evidence boosting. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2530–2535, Hyderabad, India, January 6-12 2007.

Andrew McCallum. Efficiently inducing features of conditional random fields. In Christopher Meek and Uffe Kjaerulff, editors, *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI 2003)*, pages 403–410. Morgan Kaufmann, 2003.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pages 591–598. Morgan Kaufmann, 2000.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

Andrew Y. Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2002.

Ning Qian and Terrence J. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884, 1988.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Somerset, New Jersey, 1996. Association for Computational Linguistics.

Terrence J. Sejnowski and Charles R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In Marti Hearst and Mari Ostendorf, editors, *HLT-NAACL 2003: Main Proceedings*, pages 213–220, Edmonton, Alberta, Canada, May 27 – June 1 2003. Association for Computational Linguistics.

Charles Sutton, Michael Sindelar, and Andrew McCallum. Reducing weight undertraining in structured discriminative learning. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 89–95, Morristown, NJ, USA, 2006. Association for Computational Linguistics.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 25–32. MIT Press, Cambridge, MA, 2004.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, pages 823–830, Banff, Canada, 2004. ACM Press.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. Accelerated training of conditional random fields with stochastic gradient methods. In William W. Cohen and Andrew Moore, editors, *Proceedings of the 23rd International Conference on Machine learning (ICML 2006)*, pages 969–976, New York, NY, USA, 2006. ACM.

Journal of Machine Learning Research 9 (2008) 2141-2170 Submitted 9/07; Revised 3/08; Published 10/08

HPB: A Model for Handling BN Nodes with High Cardinality Parents

Jorge Jambeiro Filho JORGE.FILHO@JAMBEIRO.COM.BR
Alfândega do Aeroporto de Viracopos
Rodovia Santos Dummont, Km 66
Campinas-SP, Brazil, CEP 13055-900

Jacques Wainer WAINER@IC.UNICAMP.BR
Instituto de Computação
Universidade Estadual de Campinas
Caixa Postal 6176
Campinas - SP, Brazil, CEP 13083-970

Editor: Bianca Zadrozny

Abstract

We replaced the conditional probability tables of Bayesian network nodes whose parents have high cardinality with a multilevel empirical hierarchical Bayesian model called hierarchical pattern Bayes (HPB).[1] The resulting Bayesian networks achieved significant performance improvements over Bayesian networks with the same structure and traditional conditional probability tables, over Bayesian networks with simpler structures like naïve Bayes and tree augmented naïve Bayes, over Bayesian networks where traditional conditional probability tables were substituted by noisy-OR gates, default tables, decision trees and decision graphs and over Bayesian networks constructed after a cardinality reduction preprocessing phase using the agglomerative information bottleneck method. Our main tests took place in important fraud detection domains, which are characterized by the presence of high cardinality attributes and by the existence of relevant interactions among them. Other tests, over UCI data sets, show that HPB may have a quite wide applicability.

Keywords: probabilistic reasoning, Bayesian networks, smoothing, hierarchical Bayes, empirical Bayes

1. Introduction

In most countries, imported goods must be declared by the importer to belong to one of large set of classes (customs codes). It is important that each good is correctly classified, because each of the customs codes implies not only different customs duties but also different administrative, sanitary, and safety requirements. The original goal of this work was to develop a tool that, considering four explanatory attributes: *declared custom code* (DCC), *importer* (IMP), *country of production* (CP) and *entry point in the receiving country* (EPR), will estimate, for each new example, the probability that it involves a misclassification. Such estimates will be used later by a larger system that allocates human resources for different types of anti-fraud operations.

Our main study data set contains 682226 examples of correct classification (which we will call negative examples) and 6460 examples of misclassification (positive examples). In this data set, the

1. This paper is a an extended version of a conference paper (Jambeiro Filho and Wainer, 2007).

first attribute assumes 7608 distinct values, the second, 18846 values, the third, 161 values, and the fourth 80 values. Thus, the domain is characterized by the presence of high cardinality attributes.

The data set is imbalanced, with only 0.93% of positive examples. This is usually handled with different resampling strategies (Chawla et al., 2002). However, resampling requires retraining the classifiers for each different assignment of costs for *false positives* and *false negatives*. In our context, such costs are not known in advance (priorities change according to other anti-fraud demands) and they vary from example to example (not all false negatives cost the same). These facts make the use of resampling techniques unattractive.

On the other hand, if we can produce reliable probability estimates directly from the original data set, the work of the human resource allocation system becomes much easier. It can at any time, define a selection rate that matches the available human resources for the specific task of detecting wrong customs codes considering all other anti-fraud demands at the moment. If the selection rate is, for example, 10%, the examples to be verified will naturally be the 10% that are most likely to involve a misclassification according to the calculated probability estimates. The allocation system may also combine the probability estimates with costs that may vary from example to example without any retraining. Thus, we decided to concentrate on Bayesian techniques.

Domain specialists claim that there are combinations of attribute values (some involving all of them) that make the probability of an instance being positive significantly higher then it could be expected looking at each value separately. They call such combinations *critical patterns*. To benefit from critical patterns we would like to use the Bayesian network (BN) (Pearl, 1988) presented in Figure 1, where all explanatory attributes are parents of the class attribute. We call a structure of this kind a *direct BN structure*.

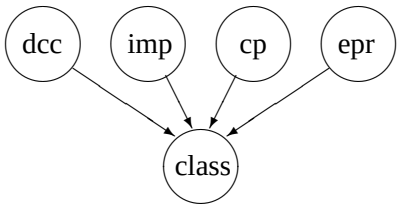


Figure 1: Direct BN structure for misclassification detection

In a BN, considering that x_{ji} is a possible value for node X_j and π_{jk} is a complete combination of values for Π_j, the set of parents of node X_j, the vector, θ_{jk}, such that $\theta_{jki} = P(x_{ji}|\pi_{jk})$ is stored in a table that is called conditional probability table (CPT) of node X_j and is assessed from the frequencies of the values of X_j among the training instances where $\Pi_j = \pi_{jk}$. The distributions of X_j given any two different combinations of values for its parents are assumed to be independent and a Dirichlet prior probability distribution for θ_{jk} is usually adopted. Applying Bayes rule and integrating over all possible values for θ_{jk} it is found that

$$E(\theta_{jki}) = P(x_{ji}|\pi_{jk}) = \frac{N_{jki} + \alpha_{jki}}{N_{jk} + \alpha_{jk}}, \tag{1}$$

where N_{jki} is the number of simultaneous observations of x_{ji} and π_{jk} in the training set, $N_{jk} = \sum_{\forall i} N_{jki}$, α_{jki} is the value of one of the parameters of the Dirichlet prior probability distribution and $\alpha_{jk} = \sum_{\forall i} \alpha_{jki}$, the equivalent sample size of the prior probability distribution.

The Dirichlet prior probability distribution is usually assumed to be noninformative, thus

$$P(x_{ji}|\pi_{jk}) = \frac{N_{jki} + \lambda}{N_{jk} + \lambda M_j}, \tag{2}$$

where all parameters of the Dirichlet distribution are equal to a small smoothing constant λ, and M_j is the number of possible values for node X_j. We call this *direct estimation* (DE). DE is sometimes called Lidstone estimate and if $\lambda = 1$ it is called Laplace estimate.

The conditional probability table of the class node of a BN with the structure in Figure 1 contains more than 1.8×10^{12} parameters. It is clear that for rarely seen combinations of attributes the choice of such structure and Equation (2) tends to produce unreliable probabilities whose calculation is dominated by the noninformative prior probability distribution.

Instead of the structure in Figure 1, we can choose a network structure that does not lead to too large tables. This can be achieved limiting the number of parents for a network node. Naïve Bayes(Duda and Hart, 1973) is an extreme example where the maximum number of parents is limited to one (the class node is the only parent of any other node). Tree augmented naïve Bayes (TAN) (Friedman et al., 1997) adds a tree to the structure of naïve Bayes connecting the explanatory attributes and limits the maximum number of parent nodes to two. However, limiting the maximum number of parents also limits the representational power of the Bayesian network(Boullé, 2005) and, thus, limits our ability to capture interactions among attributes and benefit from critical patterns. Therefore, we would prefer not to do it.

Since the high cardinality of our attributes is creating trouble, it is a reasonable idea to preprocess the data, reducing the cardinality of the attributes. We can use, for example, the agglomerative information bottleneck (AIBN) method (Slonim and Tishby, 1999) for this task. However, the process of reducing the cardinality of one attribute is blind with respect to the others (except for the class attribute) (Slonim and Tishby, 1999; Boullé, 2005; Micci-Barreca, 2001), and thus it is unlikely that cardinality reduction will result in any significant improvement in the ability to capture critical patterns, which always depend on more than one attribute.

When the number of probabilities to be estimated is too large if compared to the size of the training set and we cannot fill the traditional conditional probability tables satisfactorily, Pearl (1988) recommends the adoption of a model that resorts to causal independence assumptions like the noisy-OR gate. Using noisy-OR, the number of parameters required to represent the conditional probability distribution (CPD) of a node given its parents, instead of being proportional to the product of the cardinality of all parents attributes, becomes proportional to the sum of their cardinality. However, causal independence assumptions are incompatible with our goal of capturing critical patterns.

It is possible to use more flexible representations for the conditional probability distributions of a node given its parents, like default tables (DFs) (Friedman and Goldszmidt, 1996b), decision trees (DTs) (Friedman and Goldszmidt, 1996b) and decision graphs (DGs) (Chickering et al., 1997). According to Friedman and Goldszmidt (1996b), using such representations together with adequate learning procedures induces models that better emulate the real complexity of the interactions present in the data and the resulting network structures tend to be more complex (in terms of arcs) but require fewer parameters. Fewer parameters may result in more reliable probability estimates.

Using traditional CPTs, we assume that the probability distributions for a node given any two combinations of values for the parents are independent. If some of these distributions are actually identical, DTs, DFs and DGs, can reflect it and represent the CPD using a variable number of parameters that is only proportional to the number of actually different distributions.

On the other hand, using DTs, DFs or DGs to represent the conditional probability distributions of a node given its parents, we assume that the probability distribution of the node given two different combinations of values for the parents may be either identical or completely independent. It is possible that neither of the two assumptions hold.

Gelman et al. (2003) assert that modeling hierarchical data nonhierarchically leads to poor results. With few parameters, nonhierarchical models cannot fit the data accurately. With many parameters they fit the existing data well but lead to inferior predictions for new data. In other words they overfit the training set. In contrast, hierarchical models can fit the data well without overfitting. They can reflect similarities among distributions without assuming equality.

The slight modification in Equation (2) used by Friedman et al. (1997) in the definition of a smoothing schema for TAN shows that we can treat the data that is used to estimate a CPT as hierarchical:

$$P(x_{ji}|\pi_{jk}) = \frac{N_{jki} + S \cdot P(x_{ji})}{N_{jk} + S},$$

where S is a constant that defines the equivalent sample size of the prior probability distribution. We call this *almost direct estimation* (ADE). ADE is the consequence of adopting an informative Dirichlet prior probability distribution where $\alpha_{jki} \propto P(x_{ji})$, where $P(x_{ji})$ is the unconditional probability of x_{ji} (for the meaning of α_{jki}, see Equation 1). ADE uses the probability distribution assessed in a wider population (the whole training set) to build an informative prior probability distribution for a narrower population and so it has a hierarchical nature. In the sense of Gelman et al. (2003) ADE is an empirical hierarchical Bayesian model, not a full hierarchical Bayesian model. Probability estimation methods which use such empirical models are popularly known as empirical Bayes (EB) methods. ADE is also considered a m-estimation method (Cestnik, 1990; Zadrozny and Elkan, 2001).

We believe that ADE can get closer to the true probability distribution, but not that its discrimination power can be significantly better than DE's. It is a linear combination of two factors N_{jki}/N_{jk} and $P(x_{ji})$. The second factor is closer to the true probability distribution than its constant counterpart in *direct estimation* but it is still equal for any combination of values of Π_j and thus has no discrimination power.

ADE jumps from a very specific population (the set of training examples where $\Pi_j = \pi_{jk}$) to a very general population (the whole training set). In contrast, we present a model, that we call hierarchical pattern Bayes (HPB), which moves slowly from smaller populations to larger ones benefiting from the discrimination power available at each level.

2. Hierarchical Pattern Bayes

HPB is an empirical Bayes method that generalizes ADE into an aggressive multilevel smoothing strategy. Its name comes from the fact that it explores an hierarchy of patterns intensively, though it is not a full hierarchical Bayesian model.

Given a pattern W and a training set, D, of pairs (U_t, C_t), where U_t is the t^{th} instance in D and C_t is the class label of U_t, HPB calculates $P(C_r|W)$ for any class C_r, where a pattern is as defined below:

Definition 1 *A pattern is a set of pairs of the form* $(Attribute = Value)$*, where any attribute can appear at most once.*

An attribute that is not in the set is said to be undefined or missing. Before presenting HPB details we need a few more definitions:

Definition 2 *An instance* U *is a pair* $(iid, Pat(U))$ *where* $Pat(U)$ *is a pattern and iid is an identifier that makes each instance unique.*

Definition 3 *A pattern* Y *is more generic than a pattern* W *if and only if* $Y \subseteq W$

If Y is more generic than W, we say that W satisfies Y. If an instance U_t is such that $W = Pat(U_t)$ and W satisfies Y, we also say that U_t satisfies Y. It is worth noting that, if $Y \subseteq W$ then $S_Y \supseteq S_W$ where S_Y is the set of instances satisfying Y and S_W is the set of instances satisfying W.

Definition 4 *The level of a pattern* W*,* $level(W)$*, is the number of attributes defined in* W*.*

Definition 5 $g(W)$ *is the set of all patterns more generic than a pattern* W *whose elements have level equal to* $level(W) - 1$*.*

For example, if W is $\{A = a, B = b, C = c\}$, $g(W)$ is

$$\{\ \{B = b, C = c\}, \{A = a, C = c\}, \{A = a, B = b\}\ \}.$$

2.1 The Hierarchical Model

HPB calculates the posterior probability $P(C_r|W)$, using a strategy that is similar to almost direct estimation, but the prior probabilities are considered to be given by $P(C_r|g(W))$.

The parameters of the Dirichlet prior probability distribution used by HPB are given by $\alpha_r = S \cdot P(C_r|g(W))$, where S is a smoothing coefficient. Consequently,

$$P(C_r|W) = \frac{N_{wr} + S \cdot P(C_r|g(W))}{N_w + S}, \tag{3}$$

where N_w is the number of instances in the training set satisfying the pattern W and N_{wr} is the number of instances in the training set satisfying the pattern W whose class label is C_r.

Given Equation (3), the problem becomes to calculate $P(C_r|g(W))$. Our basic idea is to write $P(C_r|g(W))$ as a function of the various $P(C_r|W_j)$ where the W_j are patterns belonging to $g(W)$ and calculate each $P(C_r|W_j)$ recursively, using Equation (3).

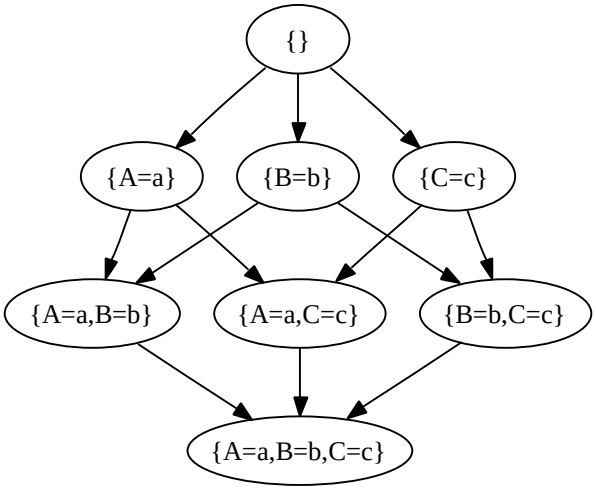


Figure 2: Example of HPB structure

Figure 2 shows a pattern hierarchy,[2] where A, B and C are the attributes. Each pattern is represented by a node and the set of parents of a pattern W in the DAG presented in Figure 2 is $g(W)$. HPB combines the posterior predictive probability distributions, $P(C_r|W_j)$, of the class given each parent, W_j, of a pattern W, to build the prior predictive probability distribution for the class given W, $P(C_r|g(W))$.

The first step to write $P(C_r|g(W))$ as a function of all the $P(C_r|W_j)$ is to apply Bayes theorem:

$$\begin{aligned} P(C_r|g(W)) &= \frac{P(g(W)|C_r)P(C_r)}{P(g(W))} \\ &\propto P(W_1, W_2, \ldots, W_L|C_r)P(C_r), \end{aligned}$$

where W_1,W_2,...,W_L are the elements of $g(W)$. Then we approximate the joint probability $P(W_1, W_2, \ldots, W_L|C_r)$ by the product of the marginal probabilities:

$$P'(C_r|g(W)) \propto P(C_r)\prod_{j=1}^{L} P(W_j|C_r), \tag{4}$$

Note that we do not assume any kind of independence when using Equation (3) to calculate posterior predictive probabilities, but we do assume independence in a naïve Bayes fashion when calculating the prior probabilities using Equation (4). Naïve Bayes is known to perform well with regard to classification error (Domingos and Pazzani, 1997) and ranking (Zhang and Su, 2004), even when its independence suppositions are violated. Assuming independence among overlapping patterns, as Equation (4) does, is equivalent to assuming independence among attributes which are known to be highly correlated, what may appear to be strange. However, naïve Bayes has been reported to perform better when attributes are highly correlated than when correlation is moderate (Rish et al., 2001).

2. Note that the DAG in Figure 2 is not a Bayesian network and the dependencies among its nodes do not follow BN conventions.

On the other hand, naïve Bayes is known to produce extreme probabilities (Domingos and Pazzani, 1997), thus we apply a calibration mechanism (Bennett, 2000; Zadrozny, 2001), which is expressed in Equation (5):

$$P''(C_r|g(W)) = (1-A)\cdot P'(C_r|g(W)) + A\cdot P(C_r), \tag{5}$$

where $A = B/(1+B)$ and B is a calibration coefficient. We discuss this calibration mechanism in Section 2.2. $P''(C_r|g(W))$ is our best estimate for $P(C_r|g(W))$ and it is used in Equation (3) as if it were the true value of $P(C_r|g(W))$.

Given Equations (4) and (5) we need to calculate $P(W_j|C_r)$. Applying Bayes theorem again,

$$P(W_j|C_r) = \frac{P(C_r|W_j)P(W_j)}{P(C_r)}. \tag{6}$$

We can estimate $P(C_r)$ is using the maximum likelihood approach: $P(C_r) = N_r/N$, where N_r is the number of examples in the training set whose class label is C_r, and N is the total number of examples in the training set. If the class variable is binary, this strategy works well, but if the class node has high cardinality it is better to employ a noninformative prior probability distribution:

$$P(C_r) = \frac{N_r + S^{NI}/M_c}{N + S^{NI}},$$

where M_c is the number of classes and S^{NI} is the smoothing constant that defines the equivalent sample size of the noninformative distribution.

When we substitute $P(W_j|C_r)$ by the right side of Equation (6) into Equation (4) we are able to clear out the factor $P(W_j)$ because it is identical for all classes:

$$\begin{aligned} P'(C_r|g(W)) &\propto P(C_r)\prod_{j=1}^{L} P(W_j|C_r) \\ &\propto P(C_r)\prod_{j=1}^{L} \frac{P(C_r|W_j)P(W_j)}{P(C_r)} \\ &\propto P(C_r)\prod_{j=1}^{L} \frac{P(C_r|W_j)}{P(C_r)}, \end{aligned}$$

so we do not need to worry about it.

Since W_j is a pattern, the estimation of $P(C_r|W_j)$ can be done recursively, using Equation (3). The recursion ends when $g(W)$ contains only the empty pattern. In this case $P(C_r|g(W)) = P(C_r|\{\{\}\}) = P(C_r)$.

2.2 Calibration Mechanism

Naïve Bayes is known to perform well in what regards to classification error (Domingos and Pazzani, 1997) and ranking (Zhang and Su, 2004), even when its independence suppositions are violated. However, naïve Bayes is also known to produce unbalanced probability estimates that are typically too "extreme" in the sense that they are too close to zero or too close to one.

The reason why naïve Bayes produces extreme probabilities is that it treats each attribute value in a pattern as if it were new information. Since attributes are not really independent, a new attribute

value is not 100% new information, treating it as if it were completely new reinforces the previous beliefs of naïve Bayes towards either zero or one. This reuse of information is explained by Bennett (2000) in the context of text classification.

In order to obtain better posterior probability distributions, calibration mechanisms which try to compensate the overly confident predictions of naïve Bayes have been proposed (Bennett, 2000; Zadrozny, 2001).

Naïve Bayes assumes that attributes are independent given the class. Equation (4) assumes that some aggregations of attributes are independent given the class. Since many of these aggregations have attributes in common, the use of Equation (4) is equivalent to assuming independence among attributes which are known to be highly correlated. Naïve Bayes has been reported to perform better when attributes are highly correlated than when correlation is moderate (Rish et al., 2001), but it is quite obvious that we are reusing a lot of information and that we can expect very extreme probability estimates. Therefore, we need to use a calibration mechanism.

Our mechanism is simpler than the ones presented by Bennett (2000) and by Zadrozny and Elkan (2002) and is unsupervised. This makes it very fast and easy to employ within each step of HPB.

We just made a linear combination of the result of Equation (4) and $P(C_r)$. We did that considering that if the estimates are more extreme than the true probabilities both near zero and near one they must match the true probabilities at some point in the middle. We believe that this point is somewhere near $P(C_r)$.

Extreme probabilities are produced when evidence in favor or against a class is reused. $P(C_r)$ is a point where either there is no evidence or there is evidence in conflicting directions in such way that the effect is null. Thus, such a point cannot be considered extreme. Our calibration mechanism attenuates the probabilities when they are extreme without affecting them in the point $P(C_r)$, where, we believe, they are already correct.

In Figure 3 we show the effect of the calibration mechanism.

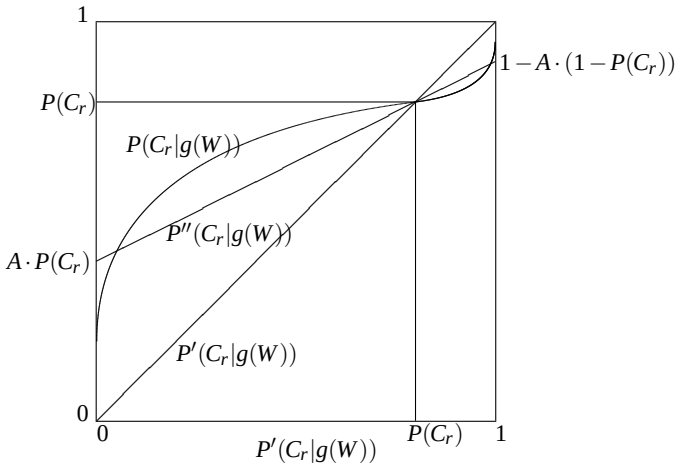


Figure 3: Effect of linear calibration over extreme probabilities

In the horizontal axis the non calibrated estimation $P'(C_r|g(W))$ is represented. The curved line represents the true probability, $P(C_r|g(W))$, as a function of $P'(C_r|g(W))$. Since all informa-

tion about $P'(C_r|g(W))$ comes from a finite data set such function never hits one or zero. When $P'(C_r|g(W))$ is near zero, $P(C_r|g(W))$ is not as near. The same happens when $P(C_r|g(W))$ is near one.

The 45^o straight line represents what would be our final estimation if we did not do any calibration, that is, $P'(C_r|g(W))$ itself. The other oblique straight line is the result of our calibration mechanism, $P''(C_r|g(W))$. It is still a linear approximation but it is much closer from $P(C_r|g(W))$ than $P'(C_r|g(W))$.

2.3 Analyzing HPB

HPB tries to explore the training set as much as possible. If there are L attributes, HPB starts its work capturing the influence of patterns of level L. At this level, all interactions among attributes may be captured as long as there are enough training instances. However, no training set is so large that we can expect that all level L patterns are well represented. Actually, if there are high cardinality attributes, it is more likely that only a minority of them are represented well. For this minority, level L dominates Equation (3) and prior probabilities are not very important. On the other hand, prior probabilities are critical for the vast majority of cases where level L patterns are not well represented in the training set. Then, HPB moves to level $L-1$. At this level, a greater fraction of patterns are well represented and it is still possible to capture the majority of attribute interactions. Many patterns of level $L-1$, however, are still not well represented and it is necessary to resort to lower level patterns. The lower are the level of the patterns the weaker is HPB's capacity to capture interactions, but less common are problems with small sample sizes.

Equation (3) combines the influence of different level patterns in a way that the most specific patterns always dominate if they are well represented. Equation (4) combines patterns in an naïve Bayes fashion, in spite of the fact that they are highly correlated. This results in extreme probability estimates that are attenuated by the calibration mechanism in Equation (5).

Since the population of instances (both in the training and in the test set) satisfying a pattern W is a subpopulation of the population of instances satisfying $W_j, \forall W_j \in g(W)$, we can say that HPB uses results previously assessed in a wider population to build informative prior probability distributions for narrower populations. Therefore, HPB is a an empirical Bayesian model, not a full hierarchical Bayesian model.

In the work of Gelman et al. (2003); Andreassen et al. (2003); Stewart et al. (2003) full hierarchical Bayesian models are presented, but they have only two levels. HPB deals with a multi level hierarchy recursively and also handles the fact that each subpopulation is contained by several overlapping superpopulations and not only by one superpopulation. These facts make it more difficult to build a full model that allows the calculation of all involved probability distributions at once considering all available evidence.

2.4 HPB as a Replacement for conditional probability tables

HPB's original goal was to be a stand alone classifier well suited a to particular domain, but it is much more relevant as a replacement for conditional probability tables.

HPB's use of space and time is exponential in the number of attributes. Thus, in domains with many attributes, it is not possible to use HPB directly. However, since the number of parents of any node in a Bayesian network is usually small because the size of a CPT is exponential in the number

of parent nodes, HPB may be used as a replacement for Bayesian networks conditional probability tables in almost any domain.

Space and time are frequently not the limiting factor for the number of parents of a BN node. More parents usually mean less reliable probabilities (Keogh and Pazzani, 1999) and it is not uncommon to limit their number to two (Friedman and Goldszmidt, 1996a; Keogh and Pazzani, 1999; Hamine and Helman, 2004). So, if HPB produces better probability estimates, it will actually allow for the addition of more parent nodes.

If the BN structure is given, the use of HPB as a replacement of the CPT of any node, X_j, is straightforward. To calculate, $P(x_{jk}|\pi_{ji})$ it is just a matter of acting as if $C_r = x_{jk}$ and $W = \pi_{ji}$, ignoring all other attributes and using HPB to calculate $P(C_r|W)$.

If the BN structure needs to be learned from data, it is necessary to choose a scoring metric that can work together with HPB in the task of choosing among the possible BN structures. We propose the use of the log-likelihood evaluated using leave-one-out cross validation:

$$LLLOO = \sum_t \log P(U_t|S, D - \{U_t\}) = \sum_t \sum_j \log P(x_{jt}|\pi_{jt}^S, D - \{U_t\}),$$

where D is the training set, U_t is the t^{th} instance of D, S is the BN structure being scored, x_{jt} is the value assumed by attribute X_j in the instance U_t, π_{jt}^S is the set of values assumed, in U_t, by the parents of X_j in S and $P(x_{jt}|\pi_{jt}^S, D - \{U_t\})$ is the value calculated by HPB for $P(x_{jt}|\pi_{jt}^S)$ using $D - \{U_t\}$ as the training set.

HPB uses the training set only through the frequencies N_{wr} and N_w in Equation (3). For fast computation of *LLLOO*, we can assess these frequencies in D and rely on the relations:

$$N_w^{D-\{U_t\}} = \begin{cases} N_w^D - 1 & \text{if } W \subset \pi_{jt}^S; \\ N_w^D & \text{otherwise;} \end{cases}$$

$$N_{wr}^{D-\{U_t\}} = \begin{cases} N_{wr}^D - 1 & \text{if } W \subset \pi_{jt}^S \wedge x_{jr} = x_{jt}; \\ N_{wr}^D & \text{otherwise.} \end{cases}$$

2.5 Selecting HPB Coefficients

Equations (3) and (5) require respectively the specifications of coefficients S and B. In the classification of a single instance, these equations are applied by HPB in the calculation of $P(C_r|W)$ for several different patterns, W. The optimal values of S and B can be different for each pattern.

In the case of the B coefficients, we use a heuristic motivated by the fact that the level of any pattern in $g(W)$ is $level(W) - 1$. The higher such level is, the more attributes in common the aggregations have, the more extreme probability estimates are and the stronger must be the effect of the calibration mechanism. Thus, we made the coefficient B in Equation (5) equal to $b(level(W) - 1)$, where b is an experimental constant.

In the case of the S coefficients, we can employ a greedy optimization approach, or, for faster training, simply define S to be a constant.

The optimization process we propose uses the area under the hit curve(Zhu, 2004) as a scoring metric. The hit curve of a classifier C over a data set D is a function, $h_{C,D}(r)$, where r is a selection rate (a real number in the interval $[0,1]$). The classifier is used to assign to each example, U_t in D the probability that U_t is a positive instance. The value of $h_{C,D}(r)$ is the number of positive instances among the $r \cdot |D|$ instances that were considered the most likely to be positive by the classifier.

We employed hit curves, instead of the more popular *Receiver Operating Characteristic Curves* (ROC) (Egan, 1975), because they match the interests of the user of a fraud detection system directly. Given a selection rate that reflects the available human resources, he/she wants to maximize the number of detected frauds.

Since the concept of a positive instance only makes sense for binary class variables, the optimization process only works for binary class problems.

When applicable, the process starts from the most general pattern family and moves toward the more specific ones, where a pattern family is the set containing all patterns that define exactly the same attributes (possibly with different values).

Assuming that the S coefficients have already been fixed for all pattern families that are more generic than a family F, there is a single S coefficient that needs to be specified to allow the use of Equation (3) to calculate $P(C_r|W)$, where W is any pattern belonging to F.

This coefficient is selected in order to maximize the area under the hit curve that is induced when, using leave-one-out cross validation, we calculate $P(C_0|W)$ for all training patterns, W, in F, where C_0 is the class that is defined to be the *positive* class.

Calculating $P(C_0|W)$ using leave-one-out cross validation, means, as explained in Section 2.4, simply subtracting one from some frequencies used by Equation (3).

2.6 Computational Complexity

The training phase of the version of HPB where constant smoothing coefficients are employed consists solely in assessing the frequencies used by Equation (3). It is easy to see that each instance, U, such that $W = Pat(U)$, in the training set, D, requires that exactly 2^L frequencies are incremented, where L is the number of parent attributes. Thus, HPB training time is

$$O(N_{tr} \cdot 2^L),$$

where N_{tr} in the number of training instances.

The test (or application) phase of HPB requires that, for each test instance, U, such that $W = Pat(U)$, the probability distribution for the class is computed given 2^L patterns. Since each computation is proportional to the number of classes, HPB test time is

$$O(N_{ts} \cdot M_c \cdot 2^L),$$

where N_{ts} in the number of test instances and M_c is the number of classes.

Note that, in both cases, HPB running time is exponential in the number of parent attributes, linear in the number of instances and independent of the cardinality of the parent attributes.

When the S coefficients are chosen by the optimization process described in Section 2.5, HPB test time does not change, but training requires that, for each pattern family, several S candidates are tested. There are 2^L pattern families and each test requires applying HPB to all training instances. Thus, HPB training time becomes

$$O(N_{tr} \cdot 2^L + N_{cand} \cdot 2^L \cdot N_{tr} \cdot M_c \cdot 2^L) = O(N_{cand} \cdot N_{tr} \cdot M_c \cdot 2^{2L}),$$

where N_{cand} is the number of candidates considered to choose a single S coefficient, which depends on the search algorithm.

HPB needs to save, for each training pattern, less than 2^L frequencies. Thus HPB use of space is

$$O(N_{tr} \cdot 2^L).$$

3. Experimental Results

We evaluated HPB in three different contexts:

- *misclassification detection*: HPB's motivation problem, an important classification problem for Brazil's Federal Revenue, where four high cardinality attributes which are supposed to have relevant interactions are used to predict a binary class attribute;
- *prediction of joint behavior*: another problem originated from Brazil's Federal Revenue where two high cardinality attributes are used to predict a third high cardinality attribute;
- *HPB as a general replacement for CPTs of Bayesian Networks*: tests over several UCI data sets comparing HPB to CPTs and other representations of the conditional probability distribution of a BN node given its parents.

In all cases the classification methods were tested using the Weka Experimenter tool (Witten and Frank, 1999) with five-fold cross validation. The machine used in the tests was an Intel Core 2 Duo 6300 with 2 GB of primary memory.

3.1 Misclassification Detection

This is the motivation problem for HPB. Considering four explanatory attributes: *declared custom code* (DCC), *importer* (IMP), *country of production* (CP) and *entry point in the receiving country* (EPR), we need to estimate, for each new example, the probability that it involves a misclassification, that is, the probability that the DCC is not the correct custom code for the goods being traded.

Our data set has 682226 examples of correct classification (which we will call negative examples) and 6460 examples of misclassification (positive examples). In this data set, the first attribute assumed 7608 distinct values, the second, 18846 values, the third, 161 values, and the fourth 80 values. There are no missing values.

We compared classifiers built using the following methods:

- *HPB-OPT*: BN with the *direct BN structure* (Figure 1), where the CPT of the class node was replaced by HPB with selection of *S* coefficients by the optimization process described in Section 2.5.
- *HPB*: BN with the *direct BN structure* (Figure 1), where the CPT of the class node was replaced by HPB with fixed *S* coefficients;
- *NB*: naïve Bayes;
- *Noisy-OR*: BN with the *direct BN structure* (Figure 1) using a noisy-OR gate instead of a CPT;
- *TAN*: Smoothed version of tree augmented naïve Bayes as described by Friedman et al. (1997) ;
- *ADE*: *almost direct estimation*. BN with the *direct BN structure*, traditional CPTs and the smoothing schema described by Friedman et al. (1997);
- *DE*: *direct estimation*. BN with the *direct BN structure* (Figure 1) and traditional CPTs;
- *DG*: Decision Graph constructed following Chickering et al. (1997). In this larger experiment, deviating from what was proposed by Chickering et al. (1997), we did not use DGs within BNs, but as standalone classification methods.

- *BN-HC-DT*: BN with decision trees learned using hill climbing (HC) and MDL as the scoring metric as described by Friedman and Goldszmidt (1996b);
- *BN-HC-DF*: BN with default tables learned using HC and MDL as described by Friedman and Goldszmidt (1996b);
- *PRIOR*: Trivial classifier that assigns the prior probability to every instance.

We were unable to build BNs with DGs replacing CPTs following Chickering et al. (1997) because it took too long (more than one day without completing a single fold). We found that the construction of a DG becomes very slow when the BN node in question has high cardinality and its parents also have high cardinality. High cardinality parents imply many possible split/merge operations to compare in each step of the learning algorithm and a high cardinality child implies that each comparison requires a lot of calculation.

In some experiments in the same domain, with BNs with DGs applied over smaller data sets, we found that in the global BN structures chosen by the search algorithm described by Chickering et al. (1997), all four explanatory attributes were parents of the class attribute. This means that if we had used a decision graph as a standalone classification method we would have had exactly the same results. Thus we concluded that it was worth to test a DG as a standalone classification method over our large data set. Since our class variable is binary the running time becomes acceptable.

We tried different parameterizations for each method and chose the parameter set that provided the best results in the five-fold cross-validation process, where best results mean best area under the hit curve up to 20% of selection rate. We ignored the area under the curve for selection rates above 20%, because all selection rates of interest are below this threshold.

Besides using the hit curve, we compared the probability distributions estimated by the models with the distribution actually found in the test set using two measures: root mean squared error (RMSE) and mean cross entropy (MCE):

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}\sum_{r=1}^{M}(P'(C_{rt}) - P(C_{rt}))^2}{MN}}, \quad MCE = \frac{\sum_{i=1}^{N}\sum_{t=1}^{M} -P(C_{rt})\log_2 P'(C_{rt})}{MN},$$

where N is the number of instances in the test set, M is the number of classes, $P'(C_{jt})$ is the estimated probability that the t^{th} instance belongs to class C_r and $P(C_{tr})$ is the true probability that t^{th} instance belongs to class C_r. $P(C_r)$ is always either 0 or 1.

Many of the methods tested require the specification of parameters and many of them are real constants. We used a common strategy to chose such constants:

1. Based on experience, decide on a search interval, $SI = [beg, end]$, within which we believe the ideal constant is;
2. Build a search enumeration SE containing all powers of 10, all halves of powers of 10 and quarters of powers of 10 within SI;
3. Try all constants in SE. If the method requires more than one constant try all possible combinations exhaustively;
4. If the optimal constant, C is in the middle of SE take C as the final constant;

5. If the optimal constant, C is one of the extreme values of SE expand SE adding one more value to it and try again. The value to be added is the real number that is the nearest to the current optimal value that was not in SE and is a power of 10, a half of a power 10 or a quarter of a power of 10.

By restricting ourselves to powers of 10, halves of powers of 10 and quarters of powers of 10 we try different orders of magnitude for the constants and avoid fine tuning them.

The smoothing coefficients employed by HPB-OPT are all automatically selected. The selection involves a leave-one-out cross validation that takes place within the current training set (the five-fold cross validation varies the current training set). The B coefficients are defined by the heuristic described in Section 2.5 and by the constant b. The choice of b was done starting with $SI = [0.5, 2.5]$. The value of S^{NI} was set to zero.

HPB requires the specification of the S constant, which is used directly and the b constant which defines the B coefficients through the heuristic in Section 2.5. The choice of b was done starting with $SI = [0.5, 2.5]$. To choose S we defined $s = S/NumClasses = S/2$ and chose s starting from $SI = [1.0, 10.0]$. The reason to introduce the constant s is just to follow the way Weka usually handles smoothing constants. Again, the value of S^{NI} was set to zero.

DGs have four parameters: the smoothing constant and three boolean parameters defining the activation state of each of the possible operations, which are complete splits, binary splits and merges. The smoothing constant was chosen starting from $SI = [0.01, 1.0]$. We always kept complete splits enabled and tried the variations resulted from enabling/disabling binary splits and merges exhaustively for each smoothing constant.

Noisy-OR and PRIOR have no parameters. The optimization of all other methods involves only the smoothing constant, which, in all cases, was chosen starting from $SI = [0.01, 2.5]$.

Below we report the optimal parameters for each method:

- *HPB-OPT*: $b = 1.0$;
- *HPB*: $s = 5.0$ and $b = 1.0$;
- *NB*: $s = 0.1$;
- *TAN*: $s = 0.25$;
- *ADE*: $s = 0.01$;
- *DE*: $s = 2.5$;
- *DG CBM*: $s = 0.05$, complete splits, binary splits and merges enabled;
- *BN-HC-DT*: $s = 0.01$;
- *BN-HC-DF*: $s = 0.025$;

In Figure 4, we show the hit curves produced by each classification method. We chose to represent the $Recall = N_{TruePositives}/N_{Positives}$, in the vertical axis, instead of the absolute number of hits, because this does not change the form of the curve and makes interpretation easier. We represented the selection rate in log scale to emphasize the beginning of the curves. In Table 2 we show the recall values for different selection rates.

In Table 1, we show the area under the hit curve (AUC), the area under the hit curve up to 20% of selection rate (AUC20), the root mean squared error (RMSE), the mean cross entropy (MCE),[3] the training time (TR) and the test time (TS) of each method. The presence of the symbol *!* before any result means that it is significantly worse than its counterpart in the first row of the table using a 95% confidence t-test. Since HPB is in the first row, we can see that HPB is significantly better than all other classifiers with regard to AUC, AUC20 and MCE. With regard to RMSE, HPB was not better than BN-HC-DT, BN-HC-DF and PRIOR.

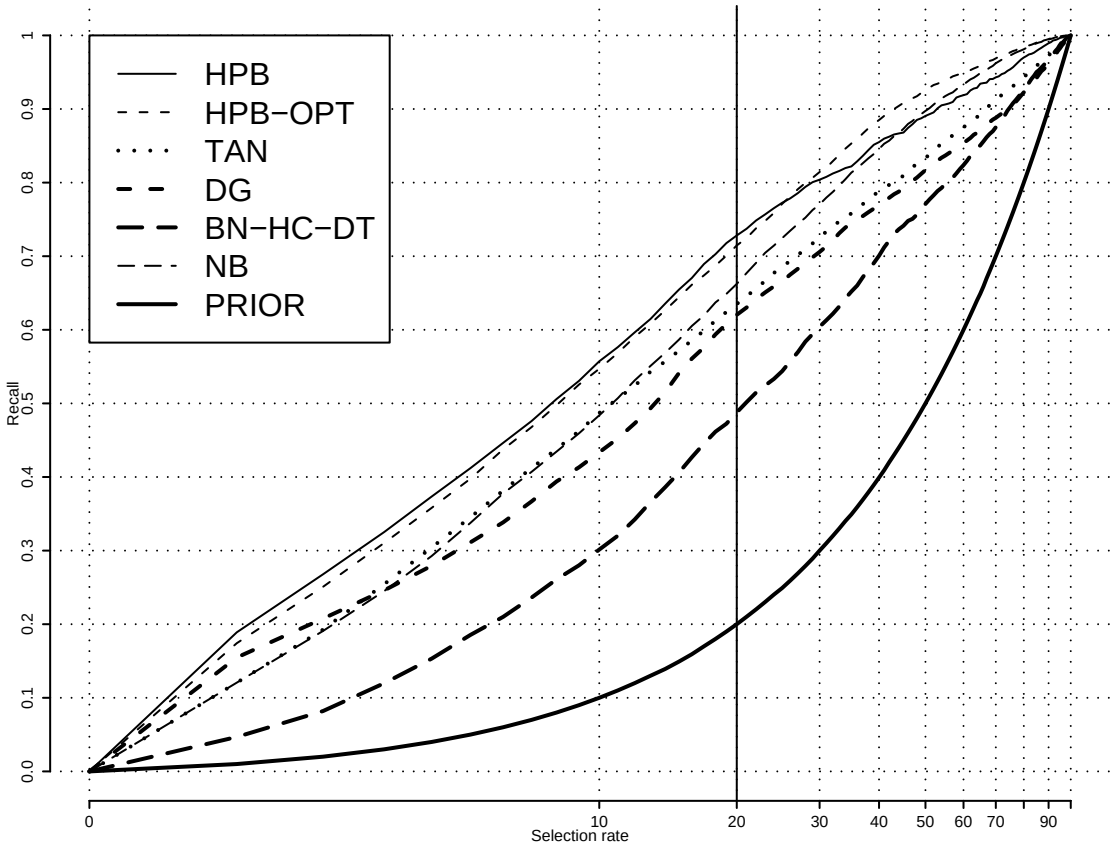


Figure 4: Misclassification detection - hit curves (to avoid pollution we only present curves related to a subset of the tested methods)

3. For better visualization of the RMSE values, MCE values and their deviations, all RMSE and MCE values presented in this paper were multiplied by 10^4.

	1%	2%	5%	10%	20%
HPB	18.89±0.77	26.77±0.57	41.20±1.10	55.72±1.82	72.81±1.69
HPB-OPT	17.41±1.55	25.08±1.10	39.76±0.61	54.70±1.44	71.45±1.74
TAN	12.06±0.59	19.26±0.70	34.52±1.32	48.70±1.82	63.52±1.06
ADE	13.32±1.37	15.06±1.46	20.70±1.65	30.61±1.18	49.39±1.06
DE	8.32±0.69	10.42±0.73	16.49±0.73	26.58±0.56	45.54±0.58
DG	15.47±1.29	20.76±0.61	31.12±1.61	43.36±2.19	62.03±1.41
BN-HC-DT	4.68±0.23	8.20±0.62	18.54±0.51	30.14±1.13	48.78±1.32
BN-HC-DF	4.44±0.39	8.22±0.49	18.45±0.44	30.06±0.30	47.45±0.98
NB	12.06±0.35	19.07±0.87	33.76±0.68	48.37±1.70	66.24±1.56
Noisy-Or	12.86±0.46	20.36±1.13	33.45±0.73	47.36±1.69	63.26±1.52
PRIOR	1.00±0.00	2.00±0.00	5.00±0.00	10.00±0.00	20.00±0.00

Table 1: Misclassification detection - other measures

	AUC	AUC20	RMSE($\times 10^4$)	MCE($\times 10^4$)	TR(s)	TS(s)
HPB	83.17±0.73	53.34±1.37	986.05±3.82	347.54±4.01	9.84±0.55	7.79±1.03
HPB-OPT	84.47±0.70	52.21±1.21	!1006.26±5.24	!367.20±5.10	!517.66±4.76	!11.43±1.50
TAN	!78.10±0.72	!45.78±1.17	!1155.36±5.26	!484.05±7.94	!43.67±0.12	1.34±0.01
ADE	!74.96±0.19	!31.43±1.25	!1005.14±6.38	!459.39±4.46	4.04±0.12	0.34±0.09
DE	!72.33±0.57	!27.37±0.40	!3462.81±2.93	!2825.06±3.79	4.35±0.11	0.28±0.00
DG	!76.12±0.90	!42.89±1.55	!1007.47±6.90	!519.49±30.82	!577.78±29.29	4.47±0.48
BN-HC-DT	!70.47±0.76	!29.95±0.85	960.89±0.25	!364.68±1.59	!125.01±1.21	!2446.17±113.19
BN-HC-DF	!69.79±0.76	!29.63±0.43	960.78±0.26	!365.03±1.25	!2433.02±20.20	!265.02±3.41
NB	!81.73±0.79	!46.33±1.08	!1120.25±6.84	!419.47±6.68	4.79±0.06	0.28±0.00
Noisy-Or	!79.13±0.64	!45.07±1.09	!1016.06±5.08	$!inf$±0.00	4.73±0.07	0.28±0.00
PRIOR	!50.48±0.01	!10.48±0.01	963.96±0.00	!383.27±0.00	4.87±0.46	0.28±0.00

Table 2: Misclassification detection - recall at different selection rates

The PRIOR method is very conservative, assigning the prior probability to every instance. In this data set, such strategy results in a good MCE and a good RMSE. On the other hand, the PRIOR method has absolutely no discrimination power, considering all instances to be equally likely to be positive. In Figure 4 and Table 2, we can see that this results in random selection, just checking that recall is always approximately equal to the selection rate.

BN-HC-DT and BN-HC-DF produced similar hit curves as can be seen in Table 2. In Figure 4 BN-HC-DT is the second worse method. The reason is that the construction of DTs and DFs presented by Friedman and Goldszmidt (1996b) turned out to be very conservative, tending to prefer simple structures: DFs with few rows and DTs with few splits. Observing the PRIOR method

results, it is not surprising that this conservative behavior results in a good MCE, a good RMSE and an unsatisfactory hit curve in comparison to other methods.

At a selection rate of 1%, ADE performs better than NB, noisy-OR and TAN, but for higher selection rates it is worse by a very significant margin. The reason is that critical patterns involving all attributes are decisive in the very beginning of the curves. ADE treats all attributes at once and thus can benefit from their presence, but soon ADE is forced to choose among test patterns for which there are no identical training patterns. At this point it starts to choose at random (in the average, 17% of the positive test instances are identical to at least one training instance).

Using Decision Graphs (with binary splits enabled), the most critical patterns were separated from the others and that resulted in a significant improvement in the beginning of the hit curve in comparison to methods like NB, noisy-OR or TAN, which cannot capture the influence of many attributes at once. However, the other patterns were clustered into few leaves in the graph. Within a leaf all patterns are considered equally likely to be positive. This resulted in loss of discrimination power for selection rates above 5%.

HPB (in both versions) benefits from critical patterns involving many or even all attributes, but also considers the influence of less specific patterns. As a consequence, it performs well for any selection rate. The version of HPB that uses a fixed value for the S coefficients is worse than NB for selection rates above 45%, but at this point, recall is already of 87% for both methods and the differences between them are never significant. Except for its non-optimized version, HPB-OPT is better than any other method for all selection rates, but the optimization process makes it fifty times slower than the simpler HPB.

It is worth noting that even the slower version of HPB is faster than the methods involving decision graphs, decision trees and default tables.

Since the cardinality of the attributes is a problem in this domain, we decided to also test all classification methods on a transformed data set where the cardinality of all attributes were reduced by the agglomerative information bottleneck method (AIBN). To prevent AIBN from using information from the test sets, we implemented a Weka meta classifier that applies AIBN immediately before training the real classifier and after each training set was separated from its associated test set in the five-fold cross validation process.

AIBN reduces the cardinality of an attribute by successively executing the merge of two values that results in minimum mutual information lost. The process can continue till a single value lasts, but can be stopped at any convenient point. We chose to limit the loss of mutual information to $1e-4$, a very low value. In spite of this, the cardinality reduction was accentuated. Table 3 shows the cardinality of the attributes before and after reduction.

Attribute	Original Cardinality	Final Cardinality
DCC	7608	101
IMP	18846	84
CP	161	50
EPR	80	28

Table 3: Cardinality reduction using AIBN

Because of the lower cardinality of the resulting attributes, it was possible to test BNs with DGs instead of standalone DGs. Results are in Table 4, Figure 5 and Table 5.

	1%	2%	5%	10%	20%
HPB	14.28±0.40	20.72±0.47	35.05±0.92	51.14±2.00	67.70±2.04
HPB-OPT	10.86±0.51	17.74±0.73	34.00±1.06	50.08±2.09	67.76±2.12
TAN	10.11±0.67	17.66±0.90	32.15±1.54	46.78±1.70	63.78±0.76
ADE	13.10±0.53	16.36±1.20	20.42±1.34	33.82±1.44	55.72±1.06
DE	8.28±0.59	11.17±0.64	19.40±0.64	32.82±0.47	56.66±0.63
BN-DG	8.14±0.46	17.40±0.66	32.12±1.38	45.44±1.12	60.66±1.48
BN-HC-DT	6.10±0.53	15.18±0.19	27.12±1.66	38.68±2.26	57.21±1.99
BN-HC-DF	6.22±0.45	14.94±0.15	26.33±0.56	37.92±1.53	55.05±1.21
NB	10.22±0.55	17.09±0.83	31.50±0.84	46.28±1.73	64.14±1.85
Noisy-Or	4.84±0.26	14.80±0.52	29.79±0.87	44.70±1.72	62.78±1.97
PRIOR	1.00±0.00	2.00±0.00	5.00±0.00	10.00±0.00	20.00±0.00

Table 4: Misclassification detection with cardinality reduction - recall at different selection rates

HPB and HPB-OPT are still the best methods but they lose much of their ability to explore critical patterns, and, at a selection rate of 1%, they do not perform nearly as well as they did over the original data set. The reason is that AIBN joins attribute values looking at each attribute separately and thus ignoring any interaction among them. In this case, relevant interactions were lost.

BNs with DGs lost much of their ability to explore critical patterns too, which also resulted in a much worse performance at a selection rate of 1%.

3.2 Prediction of Joint Behavior

In some problems of interest for Brazil's Federal Revenue it is important to answer the following question: what do two or more actors tend to do when they act together? When BNs are used to model such problems, their structure tend to follow the sketch in Figure 6.

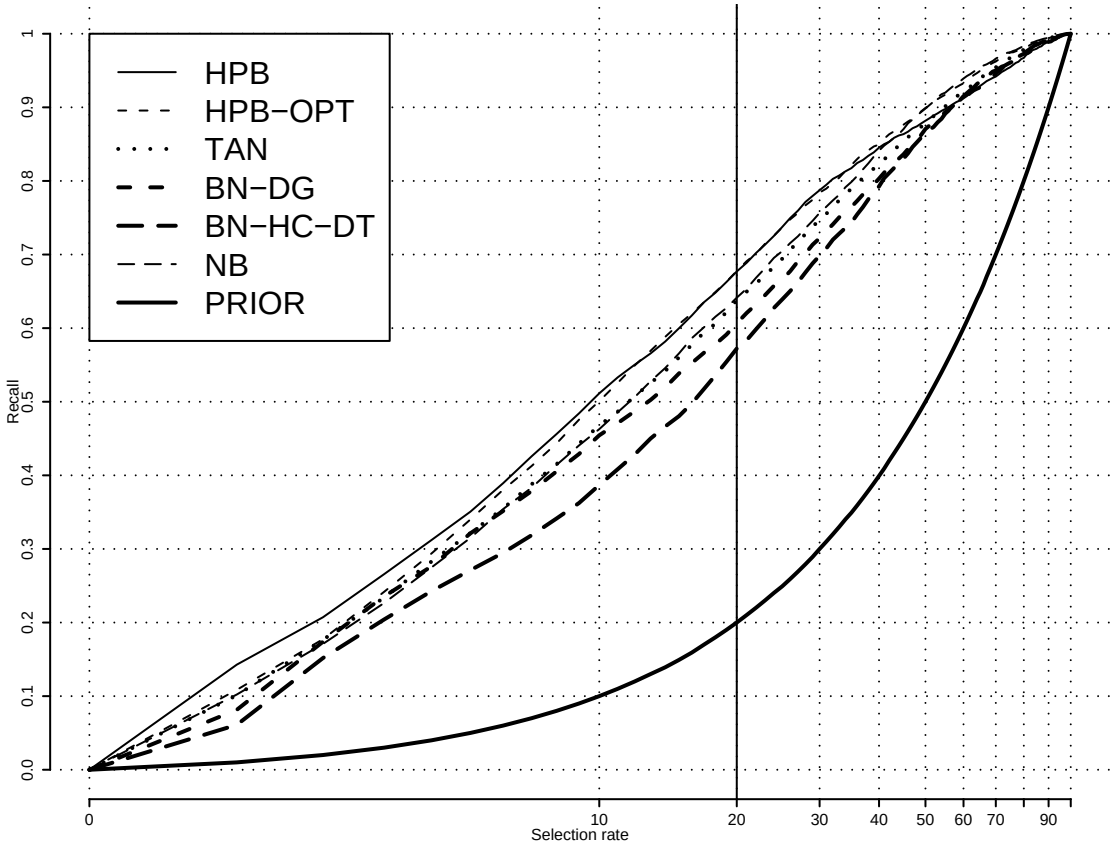


Figure 5: Misclassification detection with cardinality reduction - hit curves (to avoid pollution we only present curves related to a subset of the tested methods)

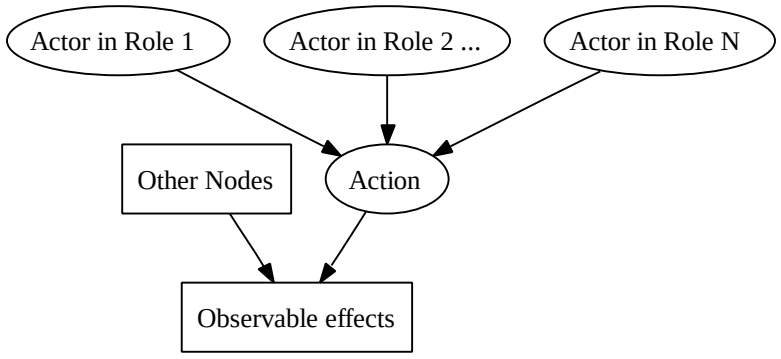


Figure 6: actors Bayesian network

	AUC	AUC20	RMSE($\times 10^4$)	MCE($\times 10^4$)	TR(s)	TS(s)
HPB	81.51±0.72	48.07±1.43	1037.82±3.51	385.10±4.59	8.30±0.07	5.74±0.03
HPB-OPT	82.16±0.85	47.28±1.44	956.09±1.06	350.67±1.98	!148.66±2.73	!6.32±0.02
TAN	!80.27±0.61	!44.21±1.15	!1103.69±5.84	!419.30±5.32	!18.40±0.53	1.40±0.02
ADE	!75.90±0.52	!35.05±1.20	953.39±1.38	354.56±2.25	2.96±0.02	0.69±0.01
DE	!75.85±0.48	!34.45±0.47	!1967.97±7.53	!914.90±6.01	!17.90±0.14	0.70±0.03
BN-DG	!78.98±0.84	!42.59±1.18	!1064.32±7.84	!393.07±8.34	!33.35±1.11	!7.84±0.09
BN-HC-DT	!77.56±0.87	!37.72±1.73	!1065.34±6.17	!393.08±1.54	!154.79±9.83	!21.37±0.80
BN-HC-DF	!77.09±0.68	!36.75±0.99	!1058.77±3.53	389.11±4.16	!234.64±56.99	!7.80±0.11
NB	81.11±0.84	!44.24±1.53	!1142.89±6.04	!429.37±6.61	!16.78±0.11	0.45±0.01
Noisy-Or	!80.11±0.84	!42.15±1.48	!1122.98±5.38	$!inf$±0.00	!16.65±0.11	0.44±0.02
PRIOR	!50.48±0.01	!10.48±0.01	963.96±0.00	383.27±0.00	!17.42±1.75	0.44±0.01

Table 5: Misclassification detection with cardinality reduction - other measures

Since the number of possible actors can be very big, but the number of roles is usually small, it seems reasonable to replace the CPT of the *Action* node in Figure 6 with HPB. However, in Section 3.1, HPB was used to predict a binary class. The number of possible actions can be high, so we have a different challenge for HPB.

In this section, we present the performance of HPB in a standalone classification problem which was built to resemble the problem of calculating the prior probability distribution of the *Action* node in Figure 6. We used two high cardinality explanatory attributes: the *importer* (IMP) and the *exporter* (EXP)[4] to predict another high cardinality variable, the *declared custom code* (DCC). Note that we are not predicting if there is a misclassification or not, but the DCC itself.

The importer attribute can assume 18846 values, the exporter attribute can assume 43880 values and the declared custom code can assume 7608 values. There are no missing values.

The tested methods were:

- *HPB*: BN with a *direct BN structure* and HPB with fixed *S* coefficients;
- *NB*: naïve Bayes;
- *ADE*: *almost direct estimation*. BN with a *direct BN structure* and the smoothing schema described by Friedman et al. (1997);
- *DE*: *direct estimation*. BN with a *direct BN structure* and traditional CPTs;
- *DE Imp*: *direct estimation*, ignoring the exporter attribute;
- *DE Exp*: *direct estimation*, ignoring the importer attribute.

HPB-OPT was not tested because its optimization process requires a binary class variable. We did not test DGs, DFs and DTs because the combination of high cardinality parents and a high cardinality child makes them too slow.

4. The *exporter* attribute was not available when we ran the tests in Section 3.1, so we did not use it there.

The parameters for each method were chosen as in Section 3.1, but MCE was used as the selection criterion. Below we present the initial search intervals and the optimal constants ($s = S/NumClasses = S/7608$):

- *HPB*: The *SI* for the *s* constant was $[1e-4, 1e-03]$ and the optimal value for *s* was equal to $1e-3$. The SI for the *b* constant was $[0.5, 2.5]$ and the optimal value for *b* was equal to 1.0. S^{NI} was always set to be equal to *S*;
- *NB*: $SI = [1e-3, 2.5]$, $s = 0.05$;
- *ADE*: $SI = [1e-3, 2.5]$, $s = 1e-3$;
- *DE*: $SI = [1e-3, 2.5]$, $s = 1e-3$;
- *DE Imp*: $SI = [1e-3, 2.5]$, $s = 1e-3$;
- *DE Exp*: $SI = [1e-3, 2.5]$, $s = 2.5e-3$;

Table 6 shows that HPB is the best method with regard to RMSE, MCE and number of correct predictions (NC). The difference is significant with the exception that HPB was not significantly better than NB with respect to number of correct assignments.

	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
HPB	108.37 ± 0.02	8.31 ± 0.00	26882.40 ± 89.76	19.51 ± 0.06	35.48 ± 0.17	1800.73 ± 2.99
DE	!108.88 ± 0.02	!9.37 ± 0.01	!25796.20 ± 63.73	!18.72 ± 0.04	1.87 ± 0.05	39.82 ± 0.03
ADE	!108.87 ± 0.01	!8.78 ± 0.01	!26039.20 ± 58.11	!18.90 ± 0.04	2.13 ± 0.01	46.17 ± 0.06
DE Exp	!108.91 ± 0.01	!9.07 ± 0.01	!25257.60 ± 64.14	!18.33 ± 0.04	2.95 ± 0.15	77.03 ± 7.94
DE Imp	!110.42 ± 0.01	!8.95 ± 0.01	!22077.60 ± 90.97	!16.02 ± 0.06	3.24 ± 0.20	73.50 ± 0.51
NB	!111.89 ± 0.05	!9.23 ± 0.01	26803.00 ± 118.12	19.45 ± 0.08	4.01 ± 0.19	357.24 ± 1.18

Table 6: Prediction of joint behavior

3.3 HPB as a General Replacement for CPTs of Bayesian Networks

In this section we test HPB over UCI data sets. Our goal is to observe its performance in domains whose characteristics are different from the ones which inspired its design. We evaluated the performance of Bayesian networks where the usual CPTs were replaced with HPB models. For comparison we also evaluated BNs using other representations for the conditional probability distribution (CPD) of a node given its parents . Below we list all tested CPD representations:

- *HPB*: HPB as described in Section 2.4;
- *DE*: direct estimation, that is, traditional CPTs;
- *ADE*: almost direct estimation. Also CPTs but using the smoothing strategy presented by Friedman et al. (1997);
- *DG*: decision graphs as presented by Chickering et al. (1997);
- *DT*: decision trees as presented by Friedman et al. (1997);
- *DF*: default tables as presented by Friedman et al. (1997).

In all cases, we learned the global BN structure using the hill climbing search algorithm implemented in Weka 3.4.2 and used NB as the starting point. To guarantee that we would not have excessively long execution times we limited the maximum number of parents to 10 and because HPB does not handle continuous attributes we removed them all. We also removed all instances with missing attributes.

Depending on the chosen representation for the CPDs, we employed different scoring metrics in the BN structure search. Below we list our choices:

- HPB: log-likelihood evaluated using leave-one-out cross validation;
- DE: MDL;
- ADE: MDL;
- DGs: Bayesian Dirichlet scoring metric as presented by Chickering et al. (1997);
- DTs: MDL as presented by Friedman and Goldszmidt (1996b);
- DFs: MDL as presented by Friedman and Goldszmidt (1996b).

The tested data sets were: *anneal*, *audiology*, *autos*, *breast-cancer*, *horse-colic*, *credit-rating*, *german-credit*, *cleveland-14-heart-disease*, *hungarian-14-heart-disease*, *hepatitis*, *hypothyroid*, *kr-vs-kp*, *labor*, *lymphography*, *mushroom*, *primary-tumor*, *sick*, *soybean*, *vote* and *zoo*.

Before building a BN we decided on fixed equivalent sample size for the prior probability distributions (this means a fixed S constant) and used it for all HPB's instances inside the BN. Fortunately, the optimal values for the equivalent sample sizes tend to be similar.

We chose S starting from $SI = [1.0, 25.0]$ and, forced the S^{NI} be identical to the S. The b constant was chosen starting from $SI = [0.5, 2.5]$.

We chose the s constant ($s = S/NumClasses$) for DGs starting from $SI = [0.01, 2.5]$. We always kept complete splits enabled and exhaustively varied the activation state of binary splits and merges. We chose the s constant for the other methods starting from $SI = [0.01, 2.5]$.

In contrast to Sections 3.1 and 3.2 we did not expand the initial search intervals if the optimal value for a constant turned out to be in one of its extreme points.

We compared the results using three criteria: number of correct classifications (NC), mean cross entropy (MCE) and root mean squared error (RMSE). To save space we present only the numbers of times where each method resulted in the best average performance. Since selecting the best parameterization for each method using a criterion and comparing the methods using only the same criterion would possibly not provide the reader enough information, we selected parameterizations using all three criteria and compared the methods using also all the three criteria in exhaustive combinations. In some cases, two or more classifiers resulted in exactly the same value for NC. In these cases, if NC was the comparison criterion, we used MCE to decide who was the winner. Results are in Table 7. Details are available in appendix A.

Sel.Crit.	Comp.Crit.	HPB	DG	DF	DT	ADE	DE
NC	NC	9	6	2	1	1	1
NC	MCE	9	5	2	1	2	1
NC	RMSE	7	6	4	1	1	1
MCE	NC	9	5	2	0	3	1
MCE	MCE	10	5	2	1	0	2
MCE	RMSE	8	6	4	0	1	1
RMSE	NC	7	7	4	0	1	1
RMSE	MCE	9	5	2	1	1	2
RMSE	RMSE	8	6	4	0	1	1

Table 7: Number of winning results in UCI data sets

HPB	DG	DF	DT	ADE	DE
3.18	3.79	1.49	1.56	1.0	1.0

Table 8: Proportions among the number of arcs of BN structures

In Table 8 we show the average proportions between the number of arcs in the BN structures learned using each CPD representation and the BN structures learned using direct estimation (traditional CPTs). We can see that, as predicted by Friedman and Goldszmidt (1996b), the use of structures like DFs, DTs and DGs does result in BNs with more arcs. The use of HPB has a similar effect.

As shown in Section 3.1, HPB is much faster than DGs, DTs and DFs in the task of handling a small set of high cardinality explanatory attributes. However, in UCI tests, many BN structures involved sets of low cardinality parents. This makes HPB comparatively slow. HPB was, in all cases, the slowest method and in some of them more than 10 times slower than the second slowest method.

Moreover, the advantage of HPB in all three criteria (MCE, RMSE and NC) was rarely statistically significant. Thus, we cannot recommend HPB as a general replacement for CPTs.

However, the vast majority of variables in the tested data sets have low cardinality (the highest cardinality variable among all data sets is the audiology class variable with 24 possible values) and many of them are binary. In spite of this, HPB's are clearly the best results in Table 7 showing that good prior probability distributions, can, many times, improve the quality of predictions.

We can say that a BN where CPDs are represented using HPB has a quite high probability of producing better classification predictions than BNs employing other CPD representations. The only explanation we found for this fact is that HPB represents some CPDs better than its alternatives and that such better representations result in BNs with better classification predictions, even when the characteristics of the attributes are opposite to the ones that inspired HPB.

This suggests that it should not be difficult to find problems where, if a BN is employed, there will be one or more BN nodes where it will be worth using HPB.

4. Conclusions

We presented HPB a novel multilevel empirical hierarchical Bayesian model, which is intended to replace conditional probability tables of Bayesian network nodes whose parents have high cardinality.

We presented HPB in two versions. The first version involves an optimization process to choose the best smoothing coefficients for each family of patterns, while the second and simpler version employs a fixed smoothing coefficient. We prefer the simpler version because it is much faster and can handle non binary child nodes.

We evaluated HPB in the domain of preselection of imported goods for human verification using hit curves, RMSE and MCE. In this domain, interactions among attributes have a great influence over the probability of finding a positive instance of misclassification, but due to the high cardinality of the attributes in this domain, exploiting such interactions is challenging.

Even the simpler version of HPB was shown capable of capturing the influence of interactions among high cardinality attributes and achieved performance improvements over standard Bayesian network methods like naïve Bayes and tree augmented naïve Bayes, over Bayesian networks where traditional conditional probability tables were substituted by noisy-OR gates, default tables, decision trees and decision graphs, and over Bayesian networks constructed after a cardinality reduction preprocessing phase using the agglomerative information bottleneck method.

HPB's execution time is exponential in the number of parents of a BN node but independent of their cardinality. Since the number of parents of a BN node is almost always small, for nodes whose parents have high cardinality, HPB, at least when its smoothing coefficients are fixed, is much faster than default tables, decision trees or decision graphs when employed to represent the conditional probability distribution of the node given its parents. This version of HPB uses the training set only through frequencies, thus data can be added dynamically without any retraining procedures other than some frequency increments.

We tested HPB in another classification problem: the prediction of the behavior of two actors when they act together. As a subproblem, this prediction is relevant in several fraud detection domains and, if the general problem is modeled as a BN, generally appears as the the task of representing the CPD of a particular node given its parents. The results, again, favored HPB.

We also provide experimental results over UCI data sets, where Bayesian network classifiers with different CPD representations are compared. Despite the fact that these data sets do not include high cardinality attributes, HPB was the representation that resulted in more winnings than any other representation in three comparison measures. The comparatively large execution times and the fact that most differences in comparison measures were not significant, do not allow us to propose HPB as a general replacement for CPTs. However, we can still conclude that BN nodes whose CPDs given their parents are better represented by HPB than by other methods are not rare. This fact indicates that HPB may have a quite wide applicability.

HPB can be very useful in practice. If specialists are handcrafting a Bayesian network structure, they want it to reflect the structure of the target problem. If this results in a node with high cardinality parents, they can just use HPB as a plug-in replacement for the CPT of the node and keep the structure they want. Without a method like HPB the high cardinality parents could easily result in unreliable probability estimates that could compromise the whole model. The specialists would have to accept a BN structure that would not reflect the target problem as closely as the original one, but which would avoid the use of high cardinality nodes as parents of the same node.

Full hierarchical Bayesian models have been widely used in the marketing community under the name of Hierarchical Bayes (Allenby et al., 1999; Lenk et al., 1996). These models have also been used in medical domains (Andreassen et al., 2003) and robotics (Stewart et al., 2003). However, we are not aware of any hierarchical Bayesian model that can replace conditional probability tables of Bayesian network nodes whose parents have high cardinality. Moreover, HPB deals with a multi level hierarchy recursively and also handles the fact that the population of instances associated to each pattern is contained by several overlapping superpopulations and not by a single one. It would be very difficult to build a full hierarchical Bayesian model that can do the same.

As future work we leave the development of better mechanisms to select HPB coefficients. Both optimization processes and heuristics should be considered. The first for the most reliable predictions and the last for fast and acceptable ones.

The pattern hierarchy employed by HPB is fixed, symmetrical (all attributes are treated the same way) and complete (all subsets of each pattern of interest are considered in the calculation of the probability of a class given the pattern). It is possible that there exists an incomplete, possibly asymmetrical hierarchy that would lead to better results. Developing an algorithm to search for such hierarchy is also left as future work.

We compared HPB against algorithms which employ the best default tables, decision trees and decision graphs chosen using some criterion. If instead of this we employed mixtures of probability tables (Fujimoto and Murata, 2006) where default tables, decision trees or decision graphs were used as category integration tables, results could be better. As a final future work we leave the development of an algorithm that can build such a mixture model and the comparison of its results to HPB's ones.

Acknowledgments

This work is part of the HARPIA project and is supported by Brazil's Federal Revenue.

Appendix A. Detailed Results over UCI Data Sets

In this appendix we detail the results of our tests over UCI data sets (see Section 3.3). To save space we only present results where the number of correct classifications (NC) was used to select the best parameterization for each method. The methods appear in the tables in decreasing order of NC. In some cases, two or more classifiers resulted in exactly the same value for NC. In these cases, we used MCE to decide the order. Results are in Table 9, Table 10 and Table 11.

ANNEAL	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-BDG	817.69 ± 123.19	199.99 ± 86.45	175.00 ± 2.54	97.43 ± 1.22	18.21 ± 0.75	0.04 ± 0.00
BN-HC-HPB	820.43 ± 69.84	182.20 ± 34.07	174.80 ± 1.09	97.32 ± 0.46	!314.35 ± 62.57	!6.24 ± 1.72
BN-HC-DF	!1289.10 ± 117.59	!469.69 ± 86.02	!168.60 ± 2.07	!93.87 ± 1.11	2.56 ± 0.08	0.02 ± 0.00
BN-HC-DT	!1486.14 ± 122.19	!610.11 ± 117.16	!166.00 ± 2.34	!92.42 ± 1.22	8.74 ± 0.36	0.04 ± 0.00
NB	!1449.24 ± 131.34	!623.72 ± 158.70	!165.00 ± 1.58	!91.87 ± 0.75	0.06 ± 0.00	0.00 ± 0.00
BN-HC-DE	!1565.84 ± 142.88	!625.42 ± 115.93	!162.60 ± 3.28	!90.53 ± 1.95	2.09 ± 0.10	0.01 ± 0.00
BN-HC-ADE	!1529.27 ± 144.05	!604.07 ± 105.71	!159.19 ± 6.14	!88.63 ± 3.28	2.09 ± 0.09	0.02 ± 0.00
AUDIOLOGY	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	1062.27 ± 164.74	443.77 ± 168.94	37.40 ± 2.19	82.75 ± 5.01	1207.20 ± 207.25	19.39 ± 4.30
NB	1199.52 ± 141.70	!1034.56 ± 355.41	35.60 ± 2.50	78.76 ± 5.56	0.05 ± 0.00	0.00 ± 0.00
BN-HC-BDG	1200.72 ± 236.04	608.81 ± 271.34	35.40 ± 4.09	78.32 ± 9.18	64.59 ± 6.16	0.06 ± 0.00
BN-HC-DF	!1343.05 ± 121.85	!728.90 ± 167.44	!32.20 ± 2.58	!71.23 ± 5.67	17.60 ± 1.97	0.04 ± 0.00
BN-HC-DE	!1521.41 ± 39.31	!906.80 ± 44.24	!28.00 ± 0.70	!61.95 ± 1.68	14.18 ± 0.44	0.03 ± 0.00
BN-HC-ADE	!1521.01 ± 39.43	!927.73 ± 51.99	!28.00 ± 0.70	!61.95 ± 1.68	14.22 ± 0.51	0.04 ± 0.00
BN-HC-DT	!1559.49 ± 49.27	!970.53 ± 52.04	!26.40 ± 1.67	!58.39 ± 3.43	13.98 ± 1.06	0.07 ± 0.00
AUTOS	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	2704.17 ± 150.42	2021.03 ± 232.97	26.40 ± 1.67	64.39 ± 4.08	1.26 ± 0.34	0.09 ± 0.01
BN-HC-BDG	2804.60 ± 156.24	!2484.20 ± 367.24	26.40 ± 1.81	64.39 ± 4.43	0.70 ± 0.07	0.00 ± 0.00
NB	2806.48 ± 157.29	2296.88 ± 433.79	!24.80 ± 0.83	!60.48 ± 2.04	0.06 ± 0.00	0.00 ± 0.00
BN-HC-DF	!3037.27 ± 123.34	!2518.42 ± 245.10	!20.60 ± 2.19	!50.24 ± 5.34	0.16 ± 0.01	0.00 ± 0.00
BN-HC-ADE	!3124.67 ± 98.05	!2682.00 ± 226.16	!18.60 ± 2.19	!45.36 ± 5.34	0.12 ± 0.00	0.00 ± 0.00
BN-HC-DE	!3124.67 ± 98.03	!2682.11 ± 226.19	!18.60 ± 2.19	!45.36 ± 5.34	0.12 ± 0.00	0.00 ± 0.00
BN-HC-DT	!3124.67 ± 98.03	!2682.11 ± 226.19	!18.60 ± 2.19	!45.36 ± 5.34	0.20 ± 0.00	0.00 ± 0.00
BREAST-CANCER	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-BDG	4401.36 ± 208.72	4164.95 ± 318.23	42.60 ± 1.81	74.47 ± 3.16	0.18 ± 0.04	0.00 ± 0.00
NB	4429.70 ± 690.84	4467.29 ± 1408.44	42.20 ± 4.32	73.79 ± 7.75	0.05 ± 0.00	0.00 ± 0.00
BN-HC-HPB	4511.98 ± 380.22	4519.62 ± 716.09	42.20 ± 2.77	73.78 ± 5.03	!0.49 ± 0.03	!0.01 ± 0.00
BN-HC-DF	4454.83 ± 217.54	4240.36 ± 327.92	41.80 ± 1.78	73.07 ± 2.96	0.14 ± 0.00	0.00 ± 0.00
BN-HC-DT	4470.69 ± 281.55	4247.27 ± 402.75	40.20 ± 4.32	70.27 ± 7.54	!0.22 ± 0.01	0.00 ± 0.00
BN-HC-DE	4485.58 ± 234.38	4284.39 ± 352.75	39.40 ± 3.78	68.88 ± 6.60	0.12 ± 0.00	0.00 ± 0.00
BN-HC-ADE	4488.31 ± 248.77	4291.26 ± 377.44	39.40 ± 3.78	68.88 ± 6.60	0.12 ± 0.00	0.00 ± 0.00
HORSE-COLIC	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	3496.97 ± 528.93	2962.44 ± 762.94	62.00 ± 3.39	84.23 ± 4.45	4.35 ± 1.80	0.05 ± 0.00
BN-HC-DT	3580.15 ± 682.03	3689.41 ± 1470.42	62.00 ± 4.41	84.23 ± 5.87	0.51 ± 0.04	0.00 ± 0.00
BN-HC-BDG	3557.76 ± 367.29	3156.62 ± 608.12	61.60 ± 3.20	83.69 ± 4.36	2.49 ± 0.48	0.00 ± 0.00
BN-HC-DF	3488.76 ± 560.79	3072.05 ± 1039.40	61.20 ± 4.08	83.15 ± 5.60	0.31 ± 0.01	0.00 ± 0.00
BN-HC-DE	3630.07 ± 699.88	3907.53 ± 1591.14	61.20 ± 3.96	83.14 ± 5.22	0.21 ± 0.00	0.00 ± 0.00
BN-HC-ADE	3648.28 ± 683.27	3900.65 ± 1545.75	60.80 ± 3.70	82.60 ± 4.92	0.21 ± 0.00	0.00 ± 0.00
NB	3951.10 ± 596.02	!4951.78 ± 1805.65	60.40 ± 3.57	82.06 ± 4.81	0.05 ± 0.00	0.00 ± 0.00
CREDIT-RATING	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-BDG	3115.11 ± 216.95	2423.92 ± 284.79	120.80 ± 3.19	87.53 ± 2.31	0.69 ± 0.17	0.00 ± 0.00
BN-HC-HPB	3245.36 ± 289.67	2578.43 ± 364.32	120.20 ± 4.14	87.10 ± 3.00	1.05 ± 0.46	!0.06 ± 0.02
BN-HC-DE	3220.08 ± 205.71	2571.82 ± 353.12	119.20 ± 2.94	86.37 ± 2.13	0.15 ± 0.00	0.00 ± 0.00
BN-HC-ADE	3221.82 ± 212.88	2577.11 ± 357.98	119.20 ± 2.94	86.37 ± 2.13	0.16 ± 0.00	0.00 ± 0.00
BN-HC-DF	3224.01 ± 122.88	2527.08 ± 166.69	118.60 ± 1.67	85.94 ± 1.21	0.19 ± 0.00	0.00 ± 0.00
BN-HC-DT	3213.24 ± 224.72	2529.39 ± 292.68	118.60 ± 3.28	85.94 ± 2.38	0.36 ± 0.02	0.00 ± 0.00
NB	3304.10 ± 195.52	2754.44 ± 278.50	118.60 ± 1.67	85.94 ± 1.21	0.06 ± 0.00	0.00 ± 0.00
GERMAN-CREDIT	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
NB	4167.54 ± 164.17	3789.33 ± 258.83	149.80 ± 7.39	74.89 ± 3.69	0.06 ± 0.00	0.00 ± 0.00
BN-HC-DF	4166.08 ± 96.36	3788.58 ± 106.86	148.60 ± 6.14	74.30 ± 3.07	!0.40 ± 0.02	!0.00 ± 0.00
BN-HC-HPB	4244.67 ± 165.79	3934.53 ± 320.17	147.80 ± 2.94	73.90 ± 1.47	!4.69 ± 1.83	!0.13 ± 0.03
BN-HC-DT	4216.53 ± 111.46	3851.92 ± 172.64	147.60 ± 4.87	73.80 ± 2.43	!1.06 ± 0.03	!0.00 ± 0.00
BN-HC-BDG	4228.17 ± 121.41	3882.09 ± 227.78	146.60 ± 5.22	73.30 ± 2.61	!1.10 ± 0.18	0.00 ± 0.00
BN-HC-ADE	4230.48 ± 126.66	3876.35 ± 199.32	145.60 ± 6.02	72.80 ± 3.01	!0.37 ± 0.01	0.00 ± 0.00
BN-HC-DE	4223.58 ± 123.27	3855.98 ± 192.26	145.40 ± 6.84	72.70 ± 3.42	!0.36 ± 0.01	0.00 ± 0.00
CLEVELAND-14-HEART-DISEASE	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-DF	2339.91 ± 113.39	1296.49 ± 119.09	49.80 ± 1.78	82.17 ± 2.72	0.09 ± 0.00	0.00 ± 0.00
BN-HC-BDG	2366.36 ± 154.85	1333.08 ± 182.49	49.60 ± 2.07	81.84 ± 3.28	!0.12 ± 0.01	0.00 ± 0.00
BN-HC-HPB	2381.80 ± 136.95	1299.65 ± 95.18	49.40 ± 1.14	81.51 ± 1.44	!0.20 ± 0.00	!0.05 ± 0.02
NB	2354.95 ± 182.24	1334.42 ± 181.36	49.40 ± 2.40	81.50 ± 3.61	0.06 ± 0.00	0.00 ± 0.00
BN-HC-DT	2362.61 ± 152.32	1338.52 ± 166.44	49.00 ± 2.54	80.84 ± 3.70	!0.13 ± 0.00	0.00 ± 0.00
BN-HC-DE	!2487.86 ± 110.45	!1495.27 ± 134.98	47.60 ± 2.70	78.54 ± 4.22	0.09 ± 0.00	0.00 ± 0.00
BN-HC-ADE	!2488.44 ± 116.62	1428.83 ± 142.14	47.40 ± 2.88	78.20 ± 4.46	0.09 ± 0.00	0.00 ± 0.00
HUNGARIAN-14-HEART-DISEASE	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	2186.65 ± 256.16	1278.47 ± 198.23	49.20 ± 2.58	83.69 ± 4.82	0.30 ± 0.05	0.04 ± 0.00
NB	2184.21 ± 410.39	1163.94 ± 327.13	48.20 ± 4.43	82.01 ± 8.05	0.05 ± 0.00	0.00 ± 0.00
BN-HC-ADE	2289.01 ± 475.44	1251.44 ± 402.26	47.60 ± 4.39	80.98 ± 7.93	0.08 ± 0.00	0.00 ± 0.00
BN-HC-DE	2299.81 ± 486.99	1275.80 ± 426.23	47.60 ± 4.39	80.98 ± 7.93	0.08 ± 0.00	0.00 ± 0.00
BN-HC-BDG	2290.08 ± 331.60	1319.64 ± 275.60	47.40 ± 3.50	80.62 ± 6.13	0.26 ± 0.02	0.00 ± 0.00
BN-HC-DF	2305.03 ± 411.21	1275.65 ± 332.30	46.60 ± 4.15	79.28 ± 7.55	0.08 ± 0.00	0.00 ± 0.00
BN-HC-DT	2330.69 ± 360.65	1278.32 ± 273.29	46.40 ± 4.87	78.94 ± 8.72	0.12 ± 0.00	0.00 ± 0.00

Table 9: Comparisons over UCI data sets

HEPATITIS	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-ADE	3337.13 ± 410.16	2722.43 ± 691.25	26.60 ± 0.89	85.80 ± 2.88	0.15 ± 0.01	0.00 ± 0.00
BN-HC-DT	3343.89 ± 668.03	2800.25 ± 1027.57	26.60 ± 1.67	85.80 ± 5.39	!0.26 ± 0.01	0.00 ± 0.00
BN-HC-DE	3369.93 ± 385.56	2767.97 ± 708.74	26.40 ± 1.51	85.16 ± 4.89	0.14 ± 0.00	0.00 ± 0.00
NB	3605.14 ± 587.26	3782.49 ± 1160.16	26.40 ± 1.51	85.16 ± 4.89	0.06 ± 0.00	0.00 ± 0.00
BN-HC-HPB	3354.12 ± 303.11	2587.67 ± 413.88	26.20 ± 0.83	84.51 ± 2.69	!0.65 ± 0.07	!0.01 ± 0.00
BN-HC-BDG	3580.20 ± 601.23	3797.96 ± 1617.52	26.20 ± 1.30	84.51 ± 4.20	!0.33 ± 0.01	0.00 ± 0.00
BN-HC-DF	3540.69 ± 455.63	3069.03 ± 1017.49	25.80 ± 0.83	83.22 ± 2.69	0.16 ± 0.01	0.00 ± 0.00
HYPOTHYROID	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	1894.74 ± 13.90	1117.79 ± 33.44	696.20 ± 0.44	92.28 ± 0.05	395.15 ± 41.22	4.80 ± 0.36
BN-HC-DF	1895.51 ± 6.60	1130.20 ± 25.31	696.20 ± 0.44	92.28 ± 0.05	4.25 ± 0.04	0.04 ± 0.00
BN-HC-ADE	1898.35 ± 7.58	1135.40 ± 16.14	696.20 ± 0.44	92.28 ± 0.05	4.25 ± 0.14	0.04 ± 0.00
BN-HC-DE	1898.35 ± 7.58	1135.44 ± 16.11	696.20 ± 0.44	92.28 ± 0.05	4.25 ± 0.12	0.03 ± 0.00
BN-HC-DT	1906.34 ± 14.12	!1159.86 ± 26.54	696.20 ± 0.44	92.28 ± 0.05	22.10 ± 0.86	0.09 ± 0.00
BN-HC-BDG	1888.97 ± 20.46	1106.31 ± 51.51	696.00 ± 0.70	92.25 ± 0.07	35.19 ± 1.43	0.07 ± 0.00
NB	1895.38 ± 11.11	1134.41 ± 31.89	696.00 ± 0.70	92.25 ± 0.11	0.06 ± 0.00	0.00 ± 0.00
KR-VS-KP	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-BDG	1301.83 ± 192.75	505.85 ± 125.79	624.60 ± 4.61	97.71 ± 0.75	161.41 ± 4.41	0.05 ± 0.00
BN-HC-HPB	1404.34 ± 203.37	574.37 ± 157.82	624.00 ± 4.74	97.62 ± 0.69	!2241.20 ± 90.05	!15.30 ± 0.95
BN-HC-DF	!1780.81 ± 276.91	!894.29 ± 172.35	!615.20 ± 8.64	!96.24 ± 1.35	20.74 ± 1.43	0.04 ± 0.00
BN-HC-DT	!1696.09 ± 153.76	!727.22 ± 133.38	!614.79 ± 2.68	!96.18 ± 0.46	!187.80 ± 11.61	!0.10 ± 0.00
BN-HC-DE	!1889.40 ± 142.39	!959.52 ± 142.96	!612.00 ± 6.00	!95.74 ± 0.90	14.04 ± 0.32	0.03 ± 0.00
BN-HC-ADE	!1888.02 ± 135.00	!931.64 ± 141.25	!611.20 ± 4.26	!95.61 ± 0.62	14.05 ± 0.34	0.03 ± 0.00
NB	!3022.02 ± 171.19	!2104.82 ± 196.02	!560.60 ± 10.23	!87.70 ± 1.54	0.06 ± 0.00	0.00 ± 0.00
LABOR	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-DT	2592.18 ± 1284.37	2068.30 ± 1725.01	10.40 ± 0.89	91.36 ± 8.73	0.09 ± 0.00	0.00 ± 0.00
BN-HC-HPB	2914.13 ± 937.77	2208.65 ± 1031.56	10.40 ± 0.89	91.36 ± 8.73	!0.26 ± 0.05	0.00 ± 0.00
BN-HC-DF	2695.25 ± 1514.59	2493.34 ± 2032.41	10.40 ± 0.89	91.36 ± 8.73	0.08 ± 0.00	0.00 ± 0.00
BN-HC-ADE	3005.58 ± 1228.36	2617.90 ± 1746.11	10.40 ± 0.89	91.36 ± 8.73	0.08 ± 0.02	0.00 ± 0.00
BN-HC-BDG	2836.04 ± 1381.80	2624.67 ± 2036.46	10.40 ± 0.89	91.36 ± 8.73	!0.14 ± 0.01	0.00 ± 0.00
BN-HC-DE	2763.72 ± 1490.36	2699.29 ± 2374.92	10.40 ± 0.89	91.36 ± 8.73	0.07 ± 0.00	0.00 ± 0.00
NB	2526.04 ± 1401.66	1968.36 ± 1567.61	10.00 ± 1.00	87.87 ± 9.65	0.06 ± 0.00	0.00 ± 0.00
LYMPHOGRAPHY	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	2512.72 ± 387.81	1824.09 ± 591.88	25.80 ± 2.16	87.12 ± 6.69	3.33 ± 1.62	0.06 ± 0.02
NB	2380.47 ± 450.03	1486.82 ± 544.14	25.60 ± 1.67	86.43 ± 4.36	0.06 ± 0.00	0.00 ± 0.00
BN-HC-BDG	2662.61 ± 659.37	2700.19 ± 1378.11	25.20 ± 2.48	85.10 ± 7.77	0.62 ± 0.12	0.00 ± 0.00
BN-HC-ADE	2754.23 ± 276.46	1944.44 ± 368.85	24.40 ± 1.94	82.36 ± 5.28	0.23 ± 0.06	0.00 ± 0.00
BN-HC-DE	2687.32 ± 344.42	1864.19 ± 420.07	24.20 ± 2.38	81.67 ± 6.86	0.18 ± 0.01	0.00 ± 0.00
BN-HC-DF	2718.45 ± 502.66	2131.58 ± 942.79	24.20 ± 1.92	81.70 ± 5.41	0.22 ± 0.01	0.00 ± 0.00
BN-HC-DT	2681.11 ± 276.46	1879.54 ± 408.83	!23.60 ± 1.14	!79.70 ± 2.65	0.41 ± 0.03	0.00 ± 0.00
MUSHROOM	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-DE	0.01 ± 0.01	0.00 ± 0.00	1624.80 ± 0.44	100.00 ± 0.00	10.50 ± 0.23	0.05 ± 0.00
BN-HC-ADE	0.01 ± 0.02	0.00 ± 0.00	1624.80 ± 0.44	100.00 ± 0.00	10.52 ± 0.26	0.05 ± 0.00
BN-HC-DF	0.08 ± 0.09	0.00 ± 0.00	1624.80 ± 0.44	100.00 ± 0.00	!14.70 ± 0.16	!0.07 ± 0.00
BN-HC-HPB	0.10 ± 0.13	0.00 ± 0.00	1624.80 ± 0.44	100.00 ± 0.00	!240.65 ± 50.90	!2.20 ± 0.20
BN-HC-BDG	!0.41 ± 0.24	!0.03 ± 0.01	1624.80 ± 0.44	100.00 ± 0.00	!76.26 ± 4.26	!0.09 ± 0.00
BN-HC-DT	28.28 ± 54.93	0.80 ± 1.36	1624.60 ± 0.54	99.98 ± 0.02	!114.42 ± 4.39	!0.22 ± 0.00
NB	!876.14 ± 56.96	!216.17 ± 36.18	!1609.00 ± 2.91	!99.02 ± 0.17	0.06 ± 0.00	0.00 ± 0.00
PRIMARY-TUMOR	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	1786.01 ± 53.39	1218.08 ± 52.72	32.60 ± 2.70	48.09 ± 4.08	11.06 ± 8.75	0.72 ± 0.04
NB	1792.85 ± 50.38	1296.85 ± 122.63	32.20 ± 2.28	47.48 ± 3.27	0.06 ± 0.00	0.00 ± 0.00
BN-HC-DF	1810.11 ± 71.59	1290.67 ± 124.59	29.80 ± 2.16	43.93 ± 2.93	0.40 ± 0.06	0.01 ± 0.00
BN-HC-BDG	!1933.29 ± 66.50	!1978.04 ± 173.90	!28.60 ± 3.50	!42.16 ± 4.98	7.59 ± 0.67	0.03 ± 0.00
BN-HC-ADE	!1983.01 ± 14.42	!1561.17 ± 33.06	!17.00 ± 0.70	!25.07 ± 1.05	0.46 ± 0.11	0.01 ± 0.00
BN-HC-DT	!2011.13 ± 2.24	!1670.29 ± 10.79	!16.80 ± 0.44	!24.78 ± 0.71	1.04 ± 0.07	0.03 ± 0.00
BN-HC-DE	!1984.37 ± 15.07	!1565.26 ± 39.21	!16.60 ± 1.14	!24.48 ± 1.71	0.47 ± 0.10	0.01 ± 0.00
SICK	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-BDG	2273.43 ± 31.56	1364.06 ± 55.35	708.40 ± 0.54	93.90 ± 0.09	34.73 ± 3.78	0.03 ± 0.00
BN-HC-HPB	2268.98 ± 21.24	1338.45 ± 23.39	708.20 ± 0.83	93.87 ± 0.10	!250.84 ± 98.29	!2.08 ± 0.85
BN-HC-ADE	2279.90 ± 16.33	1351.24 ± 43.58	708.20 ± 0.44	93.87 ± 0.05	4.15 ± 0.18	0.02 ± 0.00
BN-HC-DE	2280.38 ± 15.09	1354.35 ± 44.87	708.20 ± 0.44	93.87 ± 0.05	4.18 ± 0.13	0.01 ± 0.00
BN-HC-DF	2287.45 ± 30.30	1372.02 ± 53.92	708.20 ± 0.44	93.87 ± 0.05	4.24 ± 0.10	0.02 ± 0.00
BN-HC-DT	2283.19 ± 26.33	1346.01 ± 45.16	707.60 ± 0.89	93.79 ± 0.10	20.38 ± 0.78	!0.05 ± 0.00
NB	!2380.36 ± 50.23	!1432.11 ± 53.36	!704.60 ± 2.07	!93.39 ± 0.32	0.06 ± 0.00	0.00 ± 0.00
SOYBEAN	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-BDG	549.80 ± 52.43	75.57 ± 12.73	131.80 ± 1.30	96.48 ± 0.80	25.50 ± 1.25	0.10 ± 0.00
BN-HC-HPB	!648.62 ± 46.64	!115.11 ± 10.06	!129.40 ± 1.34	!94.72 ± 0.95	!525.54 ± 202.49	!17.23 ± 4.09
BN-HC-DF	!682.22 ± 91.60	!152.13 ± 82.49	!129.00 ± 2.34	!94.43 ± 1.60	3.42 ± 0.19	0.06 ± 0.00
NB	!730.32 ± 112.99	!267.33 ± 103.20	!128.80 ± 2.77	!94.28 ± 1.76	0.05 ± 0.00	0.00 ± 0.00
BN-HC-DE	!808.74 ± 139.12	!312.35 ± 137.43	!126.40 ± 3.28	!92.52 ± 2.17	1.49 ± 0.03	0.05 ± 0.00
BN-HC-ADE	!808.01 ± 138.81	!322.17 ± 135.10	!126.40 ± 3.28	!92.52 ± 2.17	1.48 ± 0.02	0.05 ± 0.00
BN-HC-DT	!792.47 ± 50.65	!157.82 ± 18.19	!125.20 ± 2.38	!91.65 ± 1.53	14.10 ± 0.95	!0.15 ± 0.00

Table 10: Comparisons over UCI data sets

VOTE	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	1852.76 ± 389.88	954.62 ± 345.06	83.80 ± 1.92	96.32 ± 2.21	7.96 ± 4.79	0.08 ± 0.03
BN-HC-BDG	2078.01 ± 379.00	!1806.33 ± 890.10	83.00 ± 1.58	95.40 ± 1.81	2.85 ± 0.33	0.00 ± 0.00
BN-HC-DT	1985.64 ± 632.81	1190.27 ± 623.88	82.80 ± 2.38	95.17 ± 2.74	1.82 ± 0.12	0.00 ± 0.00
BN-HC-ADE	2143.81 ± 744.24	1376.41 ± 831.41	82.20 ± 3.27	94.48 ± 3.75	0.40 ± 0.02	0.00 ± 0.00
BN-HC-DE	2119.88 ± 719.66	1417.61 ± 816.58	82.20 ± 2.58	94.48 ± 2.97	0.40 ± 0.01	0.00 ± 0.00
BN-HC-DF	2261.07 ± 621.48	1494.57 ± 821.78	81.59 ± 2.70	93.79 ± 3.10	0.46 ± 0.03	0.00 ± 0.00
NB	!2979.38 ± 540.11	!4398.98 ± 1905.74	!78.59 ± 2.60	!90.34 ± 2.99	0.05 ± 0.00	0.00 ± 0.00
ZOO	RMSE($\times 10^4$)	MCE($\times 10^4$)	NC	PC(%)	TR(s)	TS(s)
BN-HC-HPB	1076.29 ± 416.51	321.46 ± 226.17	19.20 ± 0.83	95.04 ± 3.53	1.15 ± 0.43	0.05 ± 0.01
BN-HC-DF	1000.05 ± 537.16	309.72 ± 236.93	19.00 ± 1.00	94.04 ± 4.19	0.17 ± 0.00	0.00 ± 0.00
NB	1106.38 ± 494.76	369.25 ± 238.98	19.00 ± 0.70	94.09 ± 4.07	0.05 ± 0.00	0.00 ± 0.00
BN-HC-BDG	1088.29 ± 614.67	436.05 ± 358.95	19.00 ± 1.00	94.04 ± 4.19	0.51 ± 0.04	0.00 ± 0.00
BN-HC-ADE	1007.12 ± 410.85	285.87 ± 166.44	18.80 ± 0.83	93.09 ± 4.39	0.14 ± 0.00	0.00 ± 0.00
BN-HC-DE	1108.85 ± 335.92	361.89 ± 143.52	18.80 ± 0.83	93.09 ± 4.39	0.13 ± 0.00	0.00 ± 0.00
BN-HC-DT	1303.27 ± 358.63	431.96 ± 241.95	!18.00 ± 1.00	!89.09 ± 4.21	0.40 ± 0.01	0.00 ± 0.00

Table 11: Comparisons over UCI data sets

References

Greg M. Allenby, Robert P. Leone, and Lichung Jen. A dynamic model of purchase timing with application to direct marketing. *Journal of the American Statistical Association*, 94(446):365–374, 1999.

Steen Andreassen, Brian Kristensen, Alina Zalounina, Leonard Leibovici, Uwe Frank, and Henrik C. Schonheyder. Hierarchical dirichlet learning - filling in the thin spots in a database. In Michel Dojat, Elpida T. Keravnou, and Pedro Barahona, editors, *Proceedings of the 9th Conference on Artificial Intelligence in Medicine (AIME)*, volume 2780 of *Lecture Notes in Computer Science*, pages 204–283. Springer, 2003.

Paul N. Bennett. Assessing the calibration of naive bayes' posterior estimates. Technical Report CMU-CS-00-155, School of Computer Science, Carnegie Mellon University, 2000.

Marc Boullé. A bayes optimal approach for partitioning the values of categorical attributes. *Journal of Machine Learning Research*, 6:1431–1452, 2005.

Bojan Cestnik. Estimating probabilities: a crucial task in machine learning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 147–149, 1990.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence and Research*, 16: 321–357, 2002.

David Maxwell Chickering, David Heckerman, and Christopher Meek. A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 80–89, San Franscisco, CA, 1997. Morgan Kaufman.

Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

James P. Egan. *Signal Detection Theory and Roc Analysis*. Academic Press, New York, 1975.

Nir Friedman and Moises Goldszmidt. Building classifiers using bayesian networks. In *Proceedings of the American Association for Artificial Intelligence (AAAI)/Innovative Applications of Artificial Intelligence (IAAI)*, volume 2, pages 1277–1284, 1996a.

Nir Friedman and Moises Goldszmidt. Learning bayesian networks with local structure. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Inteligence (UAI)*, pages 252–262, San Francisco, CA, 1996b. Morgan Kaufmann Publishers.

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

Yu Fujimoto and Noboru Murata. Robust estimation for mixture of probability tables based on beta-likelihood. In Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors, *Proceedings of the Sixth SIAM International Conference on Data Mining*. SIAM, 2006.

Andrew B. Gelman, John S. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall, 2. edition, 2003.

V. Hamine and P. Helman. Learning optimal augmented bayes networks. Technical Report TR-CS-2004-11, Computer Science Department, University of New Mexico, 2004.

Jorge Jambeiro Filho and Jacques Wainer. Using a hierarchical bayesian model to handle high cardinality attributes with relevant interactions in a classification problem. In *Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI)*. AAAI Press, 2007.

Eamonn J. Keogh and Michael J. Pazzani. Learning augmented bayesian classifiers: A comparison of distribution-based and classification-based approaches. In *Proceeeding of the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 225–230, Ft. Lauderdale, FL, 1999.

Peter Lenk, Wayne DeSarbo, Paul Green, and Martin Young. Hierarchical bayes conjoint analysis: recovery of part worth heterogeneity from reduced experimental designs. *Marketing Science*, 15: 173–191, 1996.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *SIGKDD Explor. Newsl.*, 3(1):27–32, 2001.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988. ISBN 1558604790.

Irina Rish, Joseph Hellerstein, and Jayram Thathachar. An analysis of data characteristics that affect naive bayes performance. Technical Report RC21993, Watson Research Center, 2001.

Noam Slonim and Naftali Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 617–623, Denver, Colorado, USA, 1999. The MIT Press. ISBN 0-262-19450-3.

Benjamin Stewart, Jonathan Ko, Dieter Fox, and Kurt Konolige. The revisiting problem in mobile robot map building: A hierarchical bayesian approach. In Christopher Meek and Uffe Kjærulff, editors, *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence (UAI)*, pages 551–558, Acapulco, Mexico, 2003. Morgan Kaufmann. ISBN 0-127-05664-5.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers Inc., 1999.

Bianca Zadrozny. Reducing multiclass to binary by coupling probability estimates. In *Proceedings of the Advances in Neural Information Processing Systems 14 (NIPS)*, Cambridge, MA, 2001. MIT Press.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 694–699. ACM Press, 2002.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 609–616, MA, USA, 2001. Morgan Kaufmann. ISBN 1-55860-778-1.

Harry Zhang and Jiang Su. Naive bayesian classifiers for ranking. *Lecture Notes in Computer Science*, 3201:501–512, 2004.

Mu Zhu. Recall, precision and average precision. Technical Report 09, Department of Statistics & Actuarial Science, University of Waterloo, 2004.

Journal of Machine Learning Research 9 (2008) 2171-2185 Submitted 6/07; Revised 1/08; Published 10/08

A Moment Bound for Multi-hinge Classifiers

Bernadetta Tarigan TARIGAN@STAT.MATH.ETHZ.CH
Sara A. van de Geer GEER@STAT.MATH.ETHZ.CH
Seminar for Statistics
Swiss Federal Institute of Technology (ETH) Zurich
Leonhardstrasse 27, 8092 Zurich, Switzerland

Editor: Peter Bartlett

Abstract

The success of support vector machines in binary classification relies on the fact that hinge loss employed in the risk minimization targets the Bayes rule. Recent research explores some extensions of this large margin based method to the multicategory case. We show a moment bound for the so-called multi-hinge loss minimizers based on two kinds of complexity constraints: entropy with bracketing and empirical entropy. Obtaining such a result based on the latter is harder than finding one based on the former. We obtain fast rates of convergence that adapt to the unknown margin.

Keywords: multi-hinge classification, all-at-once, moment bound, fast rate, entropy

1. Introduction

We consider multicategory classification with equal cost. Let $Y \in \{1,\ldots,m\}$ denote one of the m possible categories, and let $X \in \mathbb{R}^d$ be a feature. We study the classification problem, where the goal is to predict Y given X with small error. Let $\{(X_i,Y_i)\}_{i=1}^n$ be an independent and identically distributed sample from (X,Y). In the binary case ($m=2$) a classifier $f : \mathbb{R}^d \to \mathbb{R}$ can be obtained by minimizing the empirical hinge loss

$$\frac{1}{n}\sum_{i=1}^{n}(1-Y_i f(X_i))_+ \tag{1}$$

over a given class of candidate classifiers $f \in \mathcal{F}$, where $(1-Yf(X))_+ := \max(0,1-Yf(X))$ with $Y \in \{\pm 1\}$. Hinge loss in combination with a reproducing kernel Hilbert space (RKHS) regularization penalty is called the support vector machine (SVM). See, for example, Evgeniou, Pontil, and Poggio (2000). In this paper, we examine the generalization of (1) to the multicategory case ($m > 2$). We refer to this classifier as the *multi-hinge*, although, instead of RKHS-regularization we will assume a given model class $\mathcal{F}$ satisfying a complexity constraint. We show a moment bound for the excess multi-hinge risk based on two kinds of complexity constraints: entropy with bracketing and empirical entropy. Obtaining such a result based on the latter is harder than finding one based on the former. We obtain fast rates of convergence that adapt to the unknown margin.

There are two strategies to generalize the binary SVM to the multicategory SVM. One strategy is by solving a series of binary problems; the other is by considering all of the categories at once. For the first strategy, some popular methods are the one-versus-rest method and the one-versus-one method. The one-versus-rest method constructs m binary SVM classifiers. The j-th classifier f_j is trained taking the examples from class j as positive and the examples from all other categories

as negative. A new example x is assigned to the category with the largest values of $f_j(x)$. The one-versus-one method constructs one binary SVM classifier for every pair of distinct categories, that is, all together $m(m-1)/2$ binary SVM classifiers are constructed. The classifier f_{ij} is trained taking the examples from category i as positive and the examples from category j as negative. For a new example x, if f_{ij} classifies x into category i then the vote for category i is increased by one. Otherwise the vote for category j is increased by one. After each of the $m(m-1)/2$ classifiers makes its vote, x is assigned to the category with the largest number of votes. See Duan and Keerthi (2005) and the references therein for an empirical study of the performance of these methods and its variants.

An all-at-once strategy for SVM loss has been proposed by some authors. For examples, see Vapnik (2000), Weston and Watkins (1999), Crammer and Singer (2000, 2001), and Guermeur (2002). Roughly speaking, the idea is similar to the one-versus-rest approach but all the m classifiers are obtained by solving one problem. (See Hsu and Lin, 2002, for details of the formulations.) Lee, Lin, and Wahba (2004) (see also Lee, 2002) show that the relationship of the formulations of the approaches above to the Bayes' rule is not clear from the literature and that they do not always implement the Bayes' rule. They propose a new approach that has good theoretical properties. That is, the defined loss is Bayes consistent and it provides a unifying framework for both equal and unequal misclassification costs.

We consider the equal misclassification cost where a correct classification costs 0 and an incorrect classification costs 1. The target function $f : \mathbb{R}^d \to \mathbb{R}^m$ is defined as an m-tuple of separating functions with zero-sum constraint $\sum_{j=1}^m f_j(x) = 0$, for any $x \in \mathbb{R}^d$. Hence, the classifier induced by $f(\cdot)$ is

$$g(\cdot) = \arg\max_{j=1,\ldots,m} f_j(\cdot)\ . \tag{2}$$

Analogous to the binary case, when applying RKHS-regularization, each component $f_j(x)$ is considered as an element of a RKHS $\overline{\mathcal{H}}_K = \{1\} + \mathcal{H}_K$, for all $j = 1,\ldots,m$. That is, $f_j(x)$ is expressed as $h_j(x) + b_j$ with $h_j \in \mathcal{H}_K$ and b_j some constant. To find $f(\cdot) = (f_1(\cdot),\ldots,f_m(\cdot)) \in \Pi_{j=1}^m \overline{\mathcal{H}}_K$ with the zero-sum constraint, the extension of SVM methodology is to minimize

$$\frac{1}{n}\sum_{i=1}^n \sum_{j=1, j\neq Y_i}^m (f_j(X_i) + \frac{1}{m-1})_+ \ + \ \frac{\lambda}{2}\sum_{j=1}^m ||h_j||^2_{\mathcal{H}_K}\ . \tag{3}$$

Based on (3), the multi-hinge loss is now defined as

$$l(Y, f(X)) := \sum_{j=1, j\neq Y}^m (f_j(X) + \frac{1}{m-1})_+\ . \tag{4}$$

The binary SVM loss (1) is a special case by taking $m = 2$. When $Y = 1$, $l(1, f(X)) = (f_2(X) + 1)_+ = (1 - f_1(X))_+$. Similarly, when $Y = -1$, $l(-1, f(X)) = (1 + f_1(X))_+$. Thus, (4) is identical with the binary SVM loss $(1 - Y f(X))_+$, where f_1 plays the same role as f.

Using a classifier g defined as in (2), a misclassification occurs whenever $g(X) \neq Y$. Let P be the unknown underlying measure of (X,Y). The prediction error of g is $P(g(X) \neq Y)$. Let $p_j(x)$ denote the conditional probability of category j given $x \in \mathbb{R}^d$, $j = 1,\ldots,m$. The prediction error is minimized by the Bayes classifier $g^* = \arg\max_{j=1,\ldots,m} p_j$, and the smallest prediction error is $P(g^*(X) \neq Y)$.

The theoretical multi-hinge risk is the expectation of the empirical multi-hinge loss with respect to the measure P and is denoted by

$$R(f) := \int l(y, f(x))\, dP(x,y)\ , \tag{5}$$

with $l(Y, f(X))$ defined as in (4). In this setting, Bayes' rule f^* is then an m-tuple separating functions with 1 in the kth coordinate and $-1/(m-1)$ elsewhere, whenever $k = \arg\max_{j=1,\ldots,m} p_j(x)$, $x \in \mathbb{R}^d$. Lemma 1 below shows that multi-hinge loss (4) is Bayes consistent. That is, f^* minimizes multi-hinge risk (5) over all possible classifiers. We write $R^* = R(f^*)$, the smallest possible multi-hinge risk. Lemma 1 is an extension of Bayes consistency of the binary SVM that has been shown by, for example, Lin (2002), Zhang (2004a) and Bartlett, Jordan, and McAuliffe (2006).

Lemma 1. *Bayes classifier f^* minimizes the multi-hinge risk $R(f)$.*

This lemma can be found in Lee, Lin, and Wahba (2004), Zhang (2004b,c), Tewari and Bartlett (2005) and Zou, Zhu, and Hastie (2006). We give a self-contained proof in Appendix for completeness. They establish the conditions needed to achieve the consistency for a general family of multicategory loss functions extended from various large margin binary classifiers. They also show that the SVM-type losses proposed by Weston and Watkins (1999) and Crammer and Singer (2001) are not Bayes consistent. Tewari and Bartlett (2005) and Zhang (2004b,c) also show that the convergence to zero (in probability) of the excess multi-hinge risk $R(f) - R^*$ implies the convergence to zero with the same rate (in probability) of the excess prediction error $P(g(f(X)) \neq Y) - P(g(f^*(X)) \neq Y)$.

The RKHS-regularization (3) has attracted some interest. For example, Lee and Cui (2006) study an algorithm of fitting the entire regularization path and Wang and Shen (2007) study the use of l_1 penalty in place of the l_2 penalty. In this paper, we will not study the RKHS-regularization but we take the minimization of the empirical multi-hinge loss over a given class of candidate classifiers $\mathcal{F}$ satisfying a complexity constraint. That is, we do not invoke a penalization technique.

Let $\mathcal{F}$ be a model class of candidate classifiers. For $j = 1, \ldots, m$, we assume that each f_j is a member of the same class $\mathcal{F}_o = \{h : \mathbb{R}^d \to \mathbb{R}, h \in L_2(Q)\}$, with Q the unknown marginal distribution of X. That is,

$$\mathcal{F} = \{f = (f_1, \ldots, f_m) : \sum_{j=1}^{m} f_j = 0,\ f_j \in \mathcal{F}_o\}\ . \tag{6}$$

Let P_n be the empirical distribution of (X,Y) based on the observations $\{(X_i, Y_i)\}_{i=1}^n$ and Q_n the corresponding empirical distribution of X based on $X_1, \ldots, X_n$. We endow $\mathcal{F}$ with the following squared semi-metrics

$$\begin{aligned} \|f - \tilde{f}\|_{2,Q}^2 &:= \sum_{j=1}^{m} \int |f_j - \tilde{f}_j|^2\, dQ\ , \text{ and} \\ \|f - \tilde{f}\|_{2,Q_n}^2 &:= \sum_{j=1}^{m} \frac{1}{n} \sum_{i=1}^{n} |f_j(X_i) - \tilde{f}_j(X_i)|^2\ , \end{aligned}$$

for all $f, \tilde{f} \in \mathcal{F}$. We impose a complexity constraint on the class $\mathcal{F}_o$ in term of either the entropy with bracketing or the empirical entropy. Below we give the definitions of the entropies.

Definition of entropy. *Let $\mathcal{G}$ be a subset of a metric space (Λ, d). Let*

$$H(\varepsilon, \mathcal{G}, d) := \log N(\varepsilon, \mathcal{G}, d) \text{ , for all } \varepsilon > 0 \text{ ,}$$

where $N(\varepsilon, \mathcal{G}, d)$ is the smallest value of N for which there exist functions $g_1, \ldots, g_N$ in $\mathcal{G}$, such that for each $g \in \mathcal{G}$, there is a $j = j(g) \in \{1, \ldots, N\}$, such that

$$d(g, g_j) \leq \varepsilon \text{ .}$$

Then $N(\varepsilon, \mathcal{G}, d)$ is called the ε-covering number *of $\mathcal{G}$ and $H(\varepsilon, \mathcal{G}, d)$ is called the* ε-entropy *of $\mathcal{G}$ (for the d-metric).*

Definition of entropy with bracketing. *Let $\mathcal{G}$ be a subset of a metric space (Λ, d) of real-valued functions. Let*

$$H_B(\varepsilon, \mathcal{G}, d) := \log N_B(\varepsilon, \mathcal{G}, d) \text{ , for all } \varepsilon > 0 \text{ ,}$$

where $N_B(\varepsilon, \mathcal{G}, d)$ is the smallest value of N for which there exist pairs of functions $\{[g_1^L, g_1^U], \ldots, [g_N^L, g_N^U]\}$ such that $d(g_j^L, g_j^U) \leq \varepsilon$ for all $j = 1, \ldots, N$, and such that for each $g \in \mathcal{G}$, there is a $j = j(g) \in \{1, \ldots, N\}$ such that

$$g_j^L \leq g \leq g_j^U \text{ .}$$

Then $N_B(\varepsilon, \mathcal{G}, d)$ is called the ε-covering number with bracketing *of $\mathcal{G}$ and $H_B(\varepsilon, \mathcal{G}, d)$ is called the* ε-entropy with bracketing *of $\mathcal{G}$ (for the d-metric).*

Let $H_B(\varepsilon, \mathcal{F}_o, L_2(Q))$ and $H(\varepsilon, \mathcal{F}_o, L_2(Q_n))$ denote the ε-entropy with bracketing and the empirical ε-entropy of the class $\mathcal{F}_o$, respectively. The complexity of a model class can be summarized in a complexity parameter $\rho \in (0, 1)$. Let A be some positive constant. We consider classes $\mathcal{F}_o$ satisfying one of the following complexity constraints:

$$\begin{aligned} H_B(\varepsilon, \mathcal{F}_o, L_2(Q)) &\leq A\varepsilon^{-2\rho} \text{ , for all } \varepsilon > 0 \text{ , or} \\ H(\varepsilon, \mathcal{F}_o, L_2(Q_n)) &\leq A\varepsilon^{-2\rho} \text{ , for all } \varepsilon > 0 \text{ , a.s. for all } n \geq 1 \text{ .} \end{aligned}$$

It is straightforward to show that for all $\varepsilon > 0$:

$$\begin{aligned} H_B(\varepsilon, \mathcal{F}, \|\cdot\|_{2,Q}) &\leq (m-1)\, H_B(\varepsilon(m-1)^{-1/2},\ \mathcal{F}_o,\ L_2(Q)) \text{ ,} \\ H(\varepsilon, \mathcal{F}, \|\cdot\|_{2,Q_n}) &\leq (m-1)\, H(\varepsilon(2(m-1))^{-1/2},\ \mathcal{F}_o,\ L_2(Q_n)) \text{ .} \end{aligned}$$

We define the minimizer of the empirical multi-hinge loss (without penalty)

$$\hat{f}_n := \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1, j \neq Y_i}^{m} (f_j(X_i) + \frac{1}{m-1})_+ \text{ ,} \tag{7}$$

where the model class $\mathcal{F}$ defined as in (6) satisfies either an entropy with bracketing constraint or an empirical entropy constraint described above.

Besides the model class complexity, the rate of convergence also depends on the so-called margin condition (see Condition A below) that quantifies the identifiability of the Bayes rule and is summarized in a margin parameter (or noise level) $\kappa \geq 1$. In Tarigan and van de Geer (2006), a

probability inequality has been obtained for l_1-penalized excess hinge risk in the binary case that adapts to the unknown parameters. In this paper, we show a moment bound for the excess multi-hinge risk $R(\hat{f}_n) - R^*$ of $\hat{f}_n$ over the model class $\mathcal{F}$ with rate of convergence $n^{-\kappa/(2\kappa-1+\rho)}$, which is faster than $n^{-1/2}$.

In Section 2 we present our main result based on the margin and complexity conditions. The proof of the main result is given in Section 3, together with our supporting lemmas. For the sake of completeness and to avoid distraction, we place the proof of some supporting lemmas in the Appendix.

2. A Moment Bound for Multi-hinge Classifiers

We first state the margin and the complexity conditions.

Condition A (Margin condition)**.** *There exist constants $\sigma > 0$ and $\kappa \geq 1$ such that for all $f \in \mathcal{F}$,*

$$R(f) - R^* \;\geq\; \frac{1}{\sigma^\kappa}\Big(\sum_{j=1}^{m} \int |f_j - f_j^*| \, dQ\Big)^\kappa .$$

Condition B1 (Complexity constraint under ε-entropy with bracketing)**.** *Let $0 < \rho < 1$ and let A be a positive constant. The ε-entropy with bracketing satisfies the inequality*

$$H_B(\varepsilon, \mathcal{F}_o, L_2(Q)) \leq A\varepsilon^{-2\rho} \;, \;\; \textit{for all } \varepsilon > 0 \, .$$

Condition B2 (Complexity constraint under empirical ε-entropy)**.** *Let $0 < \rho < 1$ and let A be a positive constant. The empirical ε-entropy, almost surely for all $n \geq 1$, satisfies the inequality*

$$H(\varepsilon, \mathcal{F}_o, L_2(Q_n)) \leq A\varepsilon^{-2\rho} \;, \;\; \textit{for all } \varepsilon > 0 \, .$$

Now we come to the main result.

Theorem 2. *Assume Condition A is met and that $|f_j - f_j^*| \leq M$ for all $j = 1, \ldots, m$, and all $f = (f_1, \ldots, f_m) \in \mathcal{F}$. Let $\hat{f}_n$ be the multi-hinge loss minimizer defined in (7). Suppose that either Condition B1 or Condition B2 holds. Then for small values of $\delta > 0$,*

$$\mathbb{E}[R(\hat{f}_n) - R^*] \leq \frac{1+\delta}{1-\delta} \, \inf\Big\{ R(f) - R^* + C_0 \, n^{-\frac{\kappa}{2\kappa-1+\rho}} \;:\; f \in \mathcal{F} \Big\}$$

with C_0 some constant depending only on m, M, κ, σ, A and ρ.

Condition A follows from the condition on the behaviour of the conditional probabilities p_j. We formulate this in Condition AA below. We require that, for a fixed $x \in \mathbb{R}^d$, there is no pair of categories having the same conditional probabilities each of which stays away from 1. Originally

the terminology "margin condition" comes from the binary case of the prediction error considered in the work of Mammen and Tsybakov (1999) and Tsybakov (2004), where the behaviour of p_1, the conditional probability of category 1, is restricted near $\{x : p_1(x) = 1/2\}$. The "margin" set $\{x : p_1(x) = 1/2\}$ identifies the Bayes predictor which assigns a new x to class 1 if $p_1(x) > 1/2$ and class 2 otherwise. The margin condition is also called the *condition on the noise level*, and it is summarized in a margin parameter κ. Boucheron, Bousquet, and Lugosi (2005, Section 5.2) discuss the noise condition and its equivalent variants, corresponding to the fast rates of convergence, in the binary case. Thus, Condition AA is a natural extension for the multicategory case wrt. hinge loss. Lemma 3 below gives the connection between Condition A and Condition AA. We provide the proof in the Appendix. For $x \in \mathcal{X}$, let $p_k(x) = \max_{j\in\{1,\ldots,m\}} p_j(x)$ and define

$$\tau(x) := \min_{j\neq k}\{|p_j(x) - p_k(x)|, 1 - p_k(x)\}\ , \tag{8}$$

where j and k take values in $\{1,2,\ldots,m\}$.

Condition AA. *Let τ be defined in (8). There exist constants $C \geq 1$ and $\gamma \geq 0$ such that $\forall z > 0$,*

$$Q(\{\tau \leq z\}) \leq (Cz)^{1/\gamma}\ .$$

[Here we use the convention $(Cz)^{1/\gamma} = \mathbb{1}\{z \geq 1/C\}$ for $\gamma = 0$.]

Lemma 3. *Suppose Condition AA is met. Then for all $f \in \mathcal{F}$ with $|f_j - f_j^*| \leq M$ for all $j = 1,\ldots,m$,*

$$R(f) - R^* \geq \frac{1}{\sigma_M}\Big(\sum_{j=1}^m \int |f_j - f_j^*|\, dQ\Big)^{1+\gamma}\ ,$$

where $\sigma_M = C\,(mM(1/\gamma+1))^{\gamma}(1+\gamma)$. That is, Condition A holds with $\sigma = (\sigma_M)^{1/\kappa}$ and $\kappa = 1+\gamma$.

Remark. In the definition of τ we have the extra piece $1 - p_k$. It is needed for technical reason. It forces that nowhere in the input space one class can clearly dominate. We refer to the work of Bartlett and Wegkamp (2006, Section 4) and Tarigan and van de Geer (2006, Section 3.3.1) for some ideas how to get around this difficulty.

The complexity constraints B1 and B2 cover some interesting classes, including Vapnik-Chervonenkis (VC) subgraph classes and VC convex hull classes. See, for example, van der Vaart and Wellner (1996, Section 2.7), van de Geer (2000, Sections 2.4, 3.7, 7.4, 10.1 and 10.3) and Song and Wellner (2002). In the situation when the approximation error $\inf_{f\in\mathcal{F}} R(f) - R^*$ is zero (the model class $\mathcal{F}$ contains the Bayes classifier), Steinwart and Scovel (2005) obtain the same rate of convergence for the excess hinge risk under the margin condition A and the complexity condition B2. They consider the RKHS-regularization setting for the binary case instead.

We do not explore the behaviour of the approximation error $\inf_{f\in\mathcal{F}} R(f) - R^*$. This problem is still open and very hard to solve even in the binary case.

3. Proof of Theorem 2

Let $f^o := \arg\min_{f\in\mathcal{F}} R(f)$, the minimizer of the theoretical risk in the model class $\mathcal{F}$. As shorthand notation we write for the loss $l_f = l_f(X,Y) = l(Y, f(X))$. We also write $\nu_n(l_f) = \sqrt{n}\,(R_n(f) - R(f))$.

Since $R_n(\hat{f}_n) - R_n(f) \leq 0$ for all $f \in \mathcal{F}$, we have

$$\begin{aligned} R(\hat{f}_n) - R^* &\leq -[R_n(\hat{f}_n) - R(\hat{f}_n)] + [R_n(f^o) - R(f^o)] + R(f^o) - R^* \\ &\leq |\nu_n(l_{\hat{f}_n}) - \nu_n(l_{f^o})|/\sqrt{n} + R(f^o) - R^* \ . \end{aligned} \tag{9}$$

We call inequality (9) a basic-inequality, following van de Geer (2000). This upper bound enables us to work with the increments of the empirical process $\{\nu_n(l_f) - \nu_n(l_{f^o}) \ : \ l_f \in \mathcal{L}\}$ indexed by the multi-hinge loss $l_f \in \mathcal{L}$, where $\mathcal{L} = \{l_f : f \in \mathcal{F}\}$.

The procedure of the proof is based on the proof of Lemma 2.1 in del Barrio et al. (2007), page 206. We write

$$Z_n(l_f) := \frac{|\nu_n(l_f) - \nu_n(l_{f^o})|}{\left(\|l_f - l_{f^o}\|_{2,P} \vee n^{-\frac{1}{2+2\rho}}\right)^{1-\rho}} \ , \ l_f \in \mathcal{L} \ ,$$

where $(a \vee b) := \max\{a,b\}$, $\|l_f\|_{2,P}^2 := \int l_f^2(x,y)\, dP(x,y)$ and ρ is from either Condition B1 or B2. For short hand of notation, we also write $Z_n = Z_n(l_{\hat{f}_n})$. Then

$$R(\hat{f}_n) - R^* \leq (Z_n/\sqrt{n}) \ (\ \|l_{\hat{f}_n} - l_{f^o}\|_{2,P}^{1-\rho} \vee n^{-\frac{1-\rho}{2+2\rho}}) \ + \ R(f^o) - R^* \ . \tag{10}$$

Applying the triangular inequality and Lemma 4 below gives

$$\|l_{\hat{f}_n} - l_{f^o}\|_{2,P}^{1-\rho} \leq (m-1)^{(1-\rho)/2} \ \left(\|\hat{f}_n - f^*\|_{2,Q}^{1-\rho} + \|f^o - f^*\|_{2,Q}^{1-\rho}\right) .$$

Observe that for any $f \in \mathcal{F}$ with $|f_j - f_j^*| \leq M$, and for all j, Condition A gives $\|f - f^*\|_{2,Q}^2 \leq M\sigma \ (R(f) - R^*)^{1/\kappa}$. Thus,

$$\|l_{\hat{f}_n} - l_{f^o}\|_{2,P}^{1-\rho} \leq C_1 \left\{[R(\hat{f}_n) - R^*]^{(1-\rho)/2\kappa} + [R(f^o) - R^*]^{(1-\rho)/2\kappa}\right\} ,$$

with $C_1 = ((m-1)M\sigma)^{(1-\rho)/2}$. Denote by $\mathcal{R}$ the right hand side of the above inequality. Hence, from (10) we have

$$R(\hat{f}_n) - R^* \leq (Z_n/\sqrt{n}) \ (\mathcal{R} \vee n^{-\frac{1-\rho}{2+2\rho}}) + R(f^o) - R^* \ .$$

We consider first the case $(\mathcal{R} \vee n^{-\frac{1-\rho}{2+2\rho}}) = \mathcal{R}$. That is,

$$R(\hat{f}_n) - R^* \leq \frac{Z_n}{\sqrt{n}} C_1 \left\{[R(\hat{f}_n) - R^*]^{(1-\rho)/2\kappa} + [R(f^o) - R(f^*)]^{(1-\rho)/2\kappa}\right\} + R(f^o) - R^* \ .$$

Two applications of Lemma 5 below yield for all $0 < \delta < 1$,

$$\begin{aligned} R(\hat{f}_n) - R^* &\leq \delta(R(\hat{f}_n) - R^*) + (1+\delta)(R(f^o) - R^*) + 2\left(C_1 \ Z_n/\sqrt{n}\right)^{\frac{2\kappa}{2\kappa-1+\rho}} \ \delta^{-\frac{1-\rho}{2\kappa-1+\rho}} \\ &\leq \delta(R(\hat{f}_n) - R^*) \ + (1+\delta)\left(R(f^o) - R^* + C_2 \ Z_n^r \ n^{-\frac{\kappa}{2\kappa-1+\rho}}\right) , \end{aligned}$$

with $C_2 = 2 \ C_1^r \ \delta^{-\frac{1-\rho}{2\kappa-1+\rho}}$ and $r = 2\kappa/(2\kappa-1+\rho)$. Now it is left to show that $\mathbb{E}[Z_n^r]$ is bounded, say by some constant C_3. Then, $C_0 = C_2 C_3$ in Theorem 2.

To show that $\mathbb{E}[Z_n^r]$ is bounded, we use an exponential tail probability of the supremum of the weighted empirical process

$$\{Z_n(l_f) \ : \ l_f \in \mathcal{L}\} \ . \tag{11}$$

We recall that $H_B(\varepsilon, \mathcal{F}, \|\cdot\|_{2,Q}) \leq (m-1)H_B(\varepsilon(m-1)^{-1/2}, \mathcal{F}_o, L_2(Q))$. A key observation is that

$$H_B(\varepsilon, \mathcal{L}, L_2(P)) \leq (m-1)\, H_B(\varepsilon(m-1)^{-1/2}, \mathcal{F}, \|\cdot\|_{2,Q})\,,$$

by Lemma 4. It gives an upper bound for the ε-entropy with bracketing of the model class $\mathcal{L}$: $H_B(\varepsilon, \mathcal{L}, L_2(P)) \leq A_o\varepsilon^{-2\rho}$, for all $\varepsilon > 0$, with $A_o = A(m-1)^{2+2\rho}$. Under Condition B1, an application of Lemma 5.14 in van de Geer (2000), presented below in Lemma 6, gives the desired exponential tail probability. Hence, for some positive constant c,

$$\begin{aligned}\mathbb{E}[Z_n^r] &= \int_0^c \mathbb{P}(Z_n \geq t^{1/r})\, dt \,+\, \int_c^\infty \mathbb{P}(Z_n \geq t^{1/r})\, dt \\ &\leq c \,+\, \int_0^\infty c\, \exp(-\frac{t^{1/r}}{c^2})\, dt \,= c + rc^{2r+1}\Gamma(r)\,.\end{aligned}$$

For the case $\mathcal{R} \leq n^{-(1-\rho)/(2+2\rho)}$, we have

$$R(\hat{f}_n) - R^* \leq Z_n\, n^{-1/(1+\rho)} \,+\, R(f^o) - R^*\,.$$

We conclude by noting that $n^{-1/(1+\rho)} \leq n^{-\kappa/(2\kappa-1+\rho)}$, where $\kappa \geq 1$ and $0 < \rho < 1$.

Now we consider the case where Condition B2 holds instead of B1. By virtue of the proof above, we need only to verify an exponential probability of the supremum of the process (11) under Condition B2 instead of B1. This is done by employing Lemmas 7–9 below. Again, a key observation is that Lemma 4 and Condition B2 give us $H(\varepsilon, \mathcal{L}, L_2(P_n)) \leq A(m-1)^{2+2\rho}\varepsilon^{-2\rho}$. ■

Lemma 4 gives an upper bound of the squared $L_2(P)$-metric of the excess loss in terms of $\|\cdot\|_{2,Q}$-metric.

Lemma 4. $\mathbb{E}[(l_f(X,Y) - l_{f^*}(X,Y))^2] \leq (m-1)\sum_{j=1}^m \int |f_j - f_j^*|^2\, dQ\,.$

Proof. We write $\Delta(f, f^*) = \mathbb{E}_{Y|X}[(l_f(X,Y) - l_{f^*}(X,Y))^2 | X = x]$ and recall that $p_j(x) = P(Y = j | X = x)$, for all $j = 1, \ldots, m$. We fix an arbitrary $x \in \mathbb{R}^d$. Definition of the loss gives

$$\begin{aligned}\Delta(f, f^*) &= \sum_{j=1}^m p_j \Big(\sum_{i\neq j}(f_i + \frac{1}{m-1})_+ - (f_i^* + \frac{1}{m-1})_+\Big)^2 \\ &= \sum_{j=1}^m p_j \Big(\sum_{i\in I^+(j)}(f_i - f_i^*) + \sum_{i\in I^-(j)}(-\frac{1}{m-1} - f_i^*)\Big)^2\,,\end{aligned}$$

where $I^+(j) = \{i \neq j \;:\; f_i \geq -1/(m-1),\ i = 1, \ldots, m\}$ and $I^-(j) = \{i \neq j \;:\; f_i < -1/(m-1),\ i = 1, \ldots, m\}$. Use the facts that $(\sum_{i=1}^n a_i)^2 \leq n\sum_{i=1}^n a_i^2$ for all $n \in \mathbb{N}$ and $a_i \in \mathbb{R}$, and that $\max\{|I^+(j)|, |I^-(j)|\} \leq m-1$, to obtain

$$\Delta(f, f^*) \leq (m-1)\sum_{j=1}^m p_j \Big(\sum_{i\in I^+(j)}(f_i - f_i^*)^2 \,+\, \sum_{i\in I^-(j)}(-\frac{1}{m-1} - f_i^*)^2\Big)\,.$$

Clearly, $|-1/(m-1) - f_i^*| \leq |f_i - f_i^*|$ for all $i \in I^-(j)$. Hence,

$$\Delta(f, f^*) \leq (m-1)\sum_{j=1}^m p_j\, (\sum_{i\neq j}|f_i - f_i^*|^2) = (m-1)\sum_{j=1}^m (1-p_j)|f_j - f_j^*|^2\,,$$

where the last equality is obtained using $\sum_{j=1}^{m} p_j = 1$. We conclude the proof by bounding $1 - p_j$ with 1 for all j and integrating over all $x \in \mathbb{R}^d$ wrt. the marginal distribution Q. ■

The technical lemma below is an immediate consequence of Young's inequality (see, for example, Hardy, Littlewood, and Pólya, 1988, Chapter 8.3), using some straightforward bounds to simplify the expressions.

Lemma 5 (Technical Lemma)**.** *For all positive* ν, t, δ *and* $\kappa > \beta$*:*

$$\nu t^{\beta/\kappa} \;\leq\; \delta t \;+\; \nu^{\frac{\kappa}{\kappa-\beta}}\, \delta^{\frac{-\beta}{\kappa-\beta}} \;.$$

To ease the exposition, throughout Lemma 6 and Lemma 7 we write $\|\cdot\| = \|\cdot\|_{2,Q}$ and $\|\cdot\|_n = \|\cdot\|_{2,Q_n}$ for the $L_2(Q)$-norm and the $L_2(Q_n)$-norm, respectively.

Lemma 6 (van de Geer, 2000, Lemma 5.14)**.** *For a probability measure Q, let $\mathcal{H}$ be a class of uniformly bounded functions h in $L_2(Q)$, say $\sup_{h\in\mathcal{H}} |h - h^o|_\infty < 1$, where h^o is a fixed but arbitrary function in $\mathcal{H}$. Suppose that*

$$H_B(\varepsilon, \mathcal{H}, L_2(Q)) \leq A_o \varepsilon^{-2\rho} \;,\; \textit{for all } \varepsilon > 0 \;,$$

with $0 < \rho < 1$ and $A_o > 0$. Then for some positive constants c and n_o depending only on ρ and A_o,

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}} \frac{|\nu_n(h) - \nu_n(h^o)|}{\Big(\|h - h^o\| \vee n^{-\frac{1}{2+2\rho}} \Big)^{1-\rho}} \geq t \Big) \leq c\, \exp(-t/c^2) \;,$$

for all $t > c$ and $n > n_o$.

Lemma 7. *For a probability measure Q on $(\mathcal{Z}, \mathcal{A})$, let $\mathcal{H}$ be a class of uniformly bounded functions h in $L_2(Q)$, say $\sup_{h\in\mathcal{H}} |h - h^o|_\infty < 1$, where h^o is a fixed but arbitrary element in $\mathcal{H}$. Suppose that*

$$H(\varepsilon, \mathcal{H}, L_2(Q_n)) \leq A_o \varepsilon^{-2\rho} \;,\; \textit{for all } \varepsilon > 0 \;,$$

with $0 < \rho < 1$ and $A_o > 0$. Then for some positive constants c and n_o depending on ρ and A_o,

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}} \frac{|\nu_n(h) - \nu_n(h^o)|}{\Big(\|h - h^o\| \vee n^{-\frac{1}{2+2\rho}} \Big)^{1-\rho}} \geq t \Big) \leq c\, \exp(-t/c^2) \;,$$

for all $t > c$ and $n > n_o$.

Proof. For $n \geq (t^2/8)^{1+\rho/(1-\rho)}$, Chebyshev's inequality and a symmetrization technique (see, for example, van de Geer, 2000, page 32) give

$$\mathbb{P}\left(\sup_{h\in\mathcal{H}} \frac{|\nu_n(h) - \nu_n(h^o)|}{\Big(\|h - h^o\| \vee n^{-1/(2+2\rho)} \Big)^{1-\rho}} \geq t \right)$$

$$\leq 4\mathbb{P}\left(\sup_{h\in\mathcal{H}}\frac{|\nu_n^\varepsilon(h)-\nu_n^\varepsilon(h^o)|}{\left(\|h-h^o\|_n\vee n^{-1/(2+2\rho)}\right)^{1-\rho}}\geq\sqrt{t}/4\right) \tag{12}$$

$$+\,4\mathbb{P}\left(\sup_{h\in\mathcal{H}}\frac{\|h-h^o\|_n^{1-\rho}}{\left(\|h-h^o\|\vee n^{-1/(2+2\rho)}\right)^{1-\rho}}\geq\sqrt{t}/4\right), \tag{13}$$

where $\nu_n^\varepsilon(h)$ is the symmetrized version of the $\nu_n(h)$. That is, $\nu_n^\varepsilon(h)=(1/\sqrt{n})\sum_{i=1}^n\varepsilon_i h(Z_i)$, where $\{\varepsilon_i\}_{i=1}^n$ are independent random variables, independent of $\{Z_i\}_{i=1}^n$, with $\mathbb{P}(\varepsilon_i=1)=\mathbb{P}(\varepsilon_i=-1)=1/2$ for all $i=1,\ldots,n$.

To handle (12), we divide the class $\mathcal{H}$ into two disjoint classes where the empirical distance $\|h-h^o\|_n$ is smaller or larger than $n^{-1/(2+2\rho)}$. Write $\mathcal{H}_n=\{h\in\mathcal{H}:\ \|h-h^o\|_n\leq n^{-1/(2+2\rho)}\}$. By Lemma 5.1 in van de Geer (2000), stated below in Lemma 8, for some positive constant c_1,

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}_n}\frac{|\nu_n^\varepsilon(h)-\nu_n^\varepsilon(h^o)|}{n^{-(1-\rho)/(2+2\rho)}}\geq\sqrt{t}/4\Big)\leq c_1\exp\Big(-\frac{t\,n^{1/(1+\rho)}}{64\,c_1^2}\Big)\,.$$

Let $J=\min\{j>1:2^{-j}<n^{-1/(2+2\rho)}\}$. We apply the peeling device on the set $\{h\in\mathcal{H}:2^{-j}\leq\|h-h^o\|_n\leq 2^{-j+1},\ j=1,\ldots,J\}$ to obtain that, for all $t>1$,

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}_n^c}\frac{|\nu_n^\varepsilon(h)-\nu_n^\varepsilon(h^o)|}{\|h-h^o\|_n^{1-\rho}}\geq\sqrt{t}/4\ \Big|\ Z_1,\ldots,Z_n\Big)$$

$$\leq\sum_{j=1}^J\mathbb{P}\Big(\sup_{\substack{h\in\mathcal{H}\\ \|h-h^o\|_n\leq 2^{-j+1}}}|\nu_n^\varepsilon(h)-\nu_n^\varepsilon(h^o)|\geq\frac{\sqrt{t}}{4}\,2^{-j(1-\rho)}\mid Z_1,\ldots,Z_n\Big)$$

$$\leq\sum_{j=1}^J c_2\exp(-\frac{t\,2^{2\rho j}}{216\,c_2^2})\ \leq c\ \exp(-t/c^2)\,.$$

To handle (13), we use a modification of Lemma 5.6 in van de Geer (2000), stated below in Lemma 9, where we take t such that $(\sqrt{t}/4)^{1/(1-\rho)}\geq 14u$. ■

Lemma 8 (van de Geer, 2000, Lemma 5.1)**.** *Let $Z_1,\ldots,Z_n,\ldots$ be i.i.d. with distribution Q on $(\mathcal{Z},\mathcal{A})$. Let $\{\varepsilon_i\}_{i=1}^n$ be independent random variables, independent of $\{Z_i\}_{i=1}^n$, with $\mathbb{P}(\varepsilon_i=1)=\mathbb{P}(\varepsilon_i=-1)=1/2$ for all $i=1,\ldots,n$. Let $\mathcal{H}\subset L_2(Q)$ be a class of functions on $\mathcal{Z}$. Write $\nu_n^\varepsilon(h):=\frac{1}{\sqrt{n}}\sum_{i=1}^n\varepsilon_i h(Z_i)$, with $h\in\mathcal{H}$. Let*

$$\mathcal{H}(\delta):=\{h\in\mathcal{H}:\|h-h^o\|_{2,Q}\leq\delta\}\,,\quad\hat{\delta}_n:=\sup_{h\in\mathcal{H}(\delta)}\|h-h^o\|_{2,Q_n}\,,$$

where h^o is a fixed but arbitrary function in $\mathcal{H}$ and Q_n is the corresponding empirical distribution of Z based on $\{Z_i\}_{i=1}^n$. For $a\geq 8C\left(\int_{a/(32\sqrt{n})}^{\hat{\delta}_n}H^{1/2}(u,\mathcal{H},Q_n)du\ \vee\ \hat{\delta}_n\right)$, where C is some positive constant, we have

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}(\delta)}|\nu_n^\varepsilon(h)-\nu_n^\varepsilon(h^o)|\geq\frac{a}{4}\Big|Z_1,\ldots,Z_n\Big)\leq C\exp(-\frac{a^2}{64C^2\hat{\delta}_n^2})\,.$$

The following lemma is a modification of Lemma 5.6 in van de Geer (2000).

Lemma 9. *For a probability measure S on $(\mathcal{Z},\mathcal{A})$, let $\mathcal{H}$ be a class of uniformly bounded functions independent of n with $\sup_{h\in\mathcal{H}}|h|_\infty \le 1$. Suppose that almost surely for all $n \ge 1$,*

$$H(\varepsilon,\mathcal{H},L_2(S_n)) \le A_o\varepsilon^{-2\rho}, \text{ for all } \varepsilon > 0 ,$$

with $0<\rho<1$ and $A_o>0$. Then, for all n,

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}} \frac{\|h\|_{2,S_n}}{\|h\|_{2,S}\vee n^{-\frac{1}{2+2\rho}}} \ge 14u\Big) \le 4\,\exp(-u^2\, n^{\frac{\rho}{1+\rho}}) ,$$

for all $u \ge 1$.

Proof. Let $\{\delta_n\}$ be a sequence with $\delta_n\to 0$, $n\delta_n^2\to\infty$, $n\delta_n^2 \ge 2A_oH(\delta_n)$ for all n with $H(\delta_n)=\delta_n^{-2\rho}$. We apply the randomization device in Pollard (1984, page 32), as follows. Let $Z_{n+1},\dots,Z_{2n}$ be an independent copy of $Z_1,\dots,Z_n$. Let $\omega_1,\dots,\omega_n$ be independent random variables, independent of $Z_1,\dots,Z_{2n}$, with $\mathbb{P}(\omega_i=1)=\mathbb{P}(\omega_i=0)=1/2$ for all $i=1,\dots,n$. Set $Z_i{}' = Z_{2i-1+\omega_i}$ and $Z_i{}'' = Z_{2i-\omega_i}$, $i=1,\dots,n$, and $S_n{}' = (1/n)\sum_{i=1}^n \delta_{Z_i'}$, $S_n{}'' = (1/n)\sum_{i=1}^n \delta_{Z_i''}$, and $\bar{S}_{2n} = (S_n{}' + S_n{}'')/2$. Since the class is uniformly bounded by 1, an application of Chebyshev's inequality gives that for each h in $\mathcal{H}$,

$$\mathbb{P}\Big(\frac{\|h\|_{2,S_n}}{\|h\|_{2,S}\vee\delta_n} \le 2u\Big) \ge 1-\frac{1}{4u^2} \ge 3/4,$$

for all $u\ge 1$. Use a symmetrization lemma of Pollard (1984, Lemma II.3.8), see Appendix, to obtain

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}}\frac{\|h\|_{2,S_n}}{\|h\|_{2,S}\vee\delta_n}\ge 14u\Big) \le 2\mathbb{P}\Big(\sup_{h\in\mathcal{H}}\frac{|\|h\|_{2,S_n'}-\|h\|_{2,S_n''}|}{\|h\|_{2,S}\vee\delta_n}\ge 12u\Big) .$$

The peeling device on the set

$$\{h\in\mathcal{H} : (2u)^{j-1}\delta_n \le \|h\|_{2,S} \le (2u)^j\delta_n\ , j=1,2,\dots\}$$

and the inequality in Pollard (1984, page 33) give

$$\mathbb{P}\Big(\sup_{h\in\mathcal{H}}\frac{|\|h\|_{2,S_n'}-\|h\|_{2,S_n''}|}{\|h\|_{2,S}\vee\delta_n}\ge 12u \;\Big|\; Z_1,\dots,Z_n\Big)$$

$$\le \sum_{j=1}^\infty \mathbb{P}\Big(\sup_{\substack{h\in\mathcal{H}\\ \|h\|_{2,S}\le(2u)^j\delta_n}} |\|h\|_{S_n'}-\|h\|_{S_n''}| \ge 6(2u)^j\delta_n \mid Z_1,\dots,Z_n\Big)$$

$$\le \sum_{j=1}^\infty 2\exp\Big(H(\sqrt{2}(2u)^j\delta_n,\mathcal{H},\bar{S}_{2n}) - 2n(2u)^{2j}\delta_n^2\Big)$$

$$\le \sum_{j=1}^\infty 2\exp\Big(H((2u)^j\delta_n,\mathcal{H},S_n{}') + H((2u)^j\delta_n,\mathcal{H},S_n{}'') - 2n(2u)^{2j}\delta_n^2\Big)$$

$$\leq \sum_{j=1}^{\infty} 2\exp\Big(-n(2u)^{2j}\delta_n^2\Big)\,, \tag{14}$$

where the last inequality is obtained using that since $n\delta_n^2 \geq 2A_oH(\delta_n)$, also $nt^2 \geq 2A_oH(t)$ for all $t \geq \delta_n$ (here $t = (2u)^j\delta_n$). Observe that, since $(2u)^{2j} \geq (2u)^2 j > u^2 j$ for all $u \geq 1$ and $j \geq 1$, we have

$$\sum_{j=1}^{\infty} \exp(-n(2u)^{2j}\delta_n^2) \leq 2\exp(-u^2 n\delta_n^2)\,, \tag{15}$$

whenever $n\delta_n^2 > \log 2$. We finish the proof by combining (14) and (15), and taking $\delta_n = n^{-\frac{1}{2+2\rho}}$. ■

Appendix A.

Proof of Lemma 1. We write $L(f(x)) = \mathbb{E}_{Y|X}[l(Y, f(X))|X = x]$ and recall that $p_j(x) = P(Y = j|X = x)$ for all $j = 1,\ldots,m$, and that $f = (f_1,\ldots,f_m)$ with $\sum_{j=1}^m f_j = 0$. Definition (4) of the loss and the fact that $\sum_{j=1}^m p_j = 1$ give

$$L(f) = \sum_{j=1}^m p_j\Big(\sum_{k=1,\,k\neq j}^m (f_k + \frac{1}{m-1})_+\Big) = \sum_{j=1}^m (1-p_j)(f_j + \frac{1}{m-1})_+\,.$$

Let $p_k = \max_{j\in\{1,\ldots,m\}} p_j$. Here $f_j^* = -1/(m-1)$ for all $j \neq k$, and $f_k^* = 1$. Let $J^+(k) = \{j \neq k\,:\, f_j \geq -1/(m-1)\,,\ j = 1,\ldots,m\}$ and $J^-(k) = \{j \neq k\,:\, f_j < -1/(m-1)\,,\ j = 1,\ldots,m\}$. Write

$$\begin{aligned}\Delta(f) &:= L(f) - L(f^*)\\ &= \sum_{j\neq k}(1-p_j)(f_j + \frac{1}{m-1})_+ \,+\, (1-p_k)(f_k + \frac{1}{m-1})_+ \,-\, (1-p_k)(1+\frac{1}{m-1})\,.\end{aligned}$$

We first consider the case $f_k \geq -1/(m-1)$. Here,

$$\Delta(f) = (1-p_k)(f_k - 1) \,+\, \sum_{j\neq k}(1-p_j)(f_j + \frac{1}{m-1})_+\,.$$

The zero-sum constraint $\sum_{j=1}^m f_j = 0$ simply implies $f_k - 1 = -\sum_{j\neq k}(f_j + \frac{1}{m-1})$. Divide the sum into the sets $J^+(k)$ and $J^-(k)$ to obtain

$$\Delta(f) = \sum_{j\in J^+(k)} (p_k - p_j)\,(f_j + \frac{1}{m-1}) \,+\, (1-p_k)\sum_{j\in J^-(k)} |f_j + \frac{1}{m-1}|\,.$$

For the case $f_k < -1/(m-1)$, observe that

$$\frac{m}{m-1} \;=\; \sum_{j\neq k}(f_j + \frac{1}{m-1}) + f_k + \frac{1}{m-1} \;<\; \sum_{j\neq k}(f_j + \frac{1}{m-1})$$

to obtain

$$\begin{aligned}\Delta(f) &= (1-p_k)\,(-\frac{m}{m-1}) + \sum_{j\neq k}(1-p_j)(f_j + \frac{1}{m-1})_+\\ &> (p_k - 1)\sum_{j\neq k}(f_j + \frac{1}{m-1}) + \sum_{j\neq k}(1-p_j)(f_j + \frac{1}{m-1})_+\\ &= \sum_{j\in J^+(k)} (p_k - p_j)\,(f_j + \frac{1}{m-1}) \,+\, (1-p_k)\sum_{j\in J^-(k)} |f_j + \frac{1}{m-1}|\,.\end{aligned}$$

In both cases clearly $L(f)-L(f^*)$ is always non-negative since $p_k - p_j$ is non-negative for all $j \neq k$. It follows that

$$R(f)-R(f^*) = \sum_{k=1}^{m} \int (L(f)-L(f^*))\ \mathbb{1}(p_k = \max_{j=1,\ldots,m} p_j)\ dQ$$

is always non-negative, with Q the unknown marginal distribution of X. ■

Proof of Lemma 3. Let τ be defined as in (8). We write $L(f(x)) = \mathbb{E}_{Y|X}[l(Y, f(X))|X = x]$ and recall that $p_j(x) = P(Y = j|X = x)$ for all $j = 1,\ldots,m$, and that $f = (f_1,\ldots,f_m)$ with $\sum_{j=1}^m f_j = 0$. From the proof of Lemma 1, clearly

$$(L(f)-L(f^*))\ \mathbb{1}(p_k = \max_{j=1,\ldots,m} p_j) \geq \tau \sum_{j\neq k} |f_j - f_j^*| \geq \frac{\tau}{2} \sum_{j=1}^{m} |f_j - f_j^*|\ ,$$

where the second inequality is obtained from the fact that $|f_k - f_k^*| \leq \sum_{j\neq k} |f_j - f_j^*|$. That is, the excess risk is lower bounded by

$$\frac{1}{2} \sum_{j=1}^{m} \int \tau |f_j - f_j^*| dQ\ .$$

It implies that, for all $z > 0$,

$$R(f) - R^* \geq \frac{z}{2} \sum_{j=1}^{m} \left[\int |f_j - f_j^*| dQ - \int_{\tau \leq z} |f_j - f_j^*| dQ \right]\ .$$

Since $|f_j - f_j^*| \leq M$ for all j, and by Condition AA, the second integral in the inequality above can be upper bounded by $M(Cz)^{1/\gamma}$. Thus, for all $z > 0$,

$$R(f) - R^* \geq \frac{z}{2} \sum_{j=1}^{m} \int |f_j - f_j^*| dQ\ -\ \frac{z}{2} m M (Cz)^{1/\gamma}\ .$$

We take $z = \left(\sum_{j=1}^m \int |f_j - f_j^*| dQ \right)^\gamma / \left(mMC^{1/\gamma}(1+\gamma^{-1}) \right)^\gamma$ when $\gamma > 0$, and $z \uparrow 1/C$ when $\gamma = 0$. ■

Symmetrization lemma (Pollard, 1984, Lemma II.3.8)**.** *Let $\{Z(t) : t \in T\}$ and $\{Z'(t) : t \in T\}$ be independent stochastic process sharing an index set T. Suppose there exist constants $\beta > 0$ and $\alpha > 0$ such that $\mathbb{P}(|Z(t)| \leq \alpha) \geq \beta$ for every $t \in T$. Then*

$$\mathbb{P}\left(\sup_t |Z(t)| > \varepsilon \right) \leq \beta^{-1} \mathbb{P}\left(\sup_t\ |Z(t) - Z'(t)| > \varepsilon - \alpha \right)\ .$$

References

Peter L. Bartlett and Marten H. Wegkamp. Classification with a reject option using a hinge loss. Technical report, U.C. Berkeley, 2006.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Stéphane Boucheron, Olivier Bousquet, and Gábor Lugosi. Theory of classification: a survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

Koby Crammer and Yoram Singer. On the learnability and design of output codes for multiclass problems. In *Proceeding of the 13th Annual Conference on Computational Learning Theory*, pages 35–46. Morgan Kaufmann, 2000.

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Eustasio del Barrio, Paul Deheuvels, and Sara A. van de Geer. *Lectures on Empirical Processes*. EMS Series of Lectures in Mathematics. European Mathematical Society, 2007.

Kaibo Duan and S. Sathiya Keerthi. Which is the best multiclass svm method? an empirical study. In *Multiple Classifier Systems*, number 3541 in Lecture Notes in Computer Science, pages 278–285. Springer Berlin/Heidelberg, 2005.

Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

Yann Guermeur. Combining discriminant models with new multiclass svms. *Pattern Analysis & Applications*, 5:168–179, 2002.

Godfrey H. Hardy, John E. Littlewood, and George Pólya. *Inequalities*. Cambridge University Press, second edition, 1988.

Chih-Wei Hsu and Chih-Jen Lin. A comparison methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

Yoonkyung Lee. *Multicategory Support Vector Machines, Theory and Application to the Classification of Microarray Data and Satellite Radiance Data*. PhD thesis, University of Wisconsin-Madison, Departement of Statistics, 2002.

Yoonkyung Lee and Zhenhuan Cui. Characterizing the solution path of multicategory support vector machines. *Statistica Sinica*, 16(2):391–409, 2006.

Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

Yi Lin. Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 6(3):259–275, 2002.

Enno Mammen and Alexandre B. Tsybakov. Smooth discrimination analysis. *Ann. Statist.*, 27(6): 1808–1829, 1999.

David Pollard. *Convergence of Stochastic Processes*. Springer-Verlag New York Inc., 1984.

Shuguang Song and Jon A. Wellner. An upper bound for uniform entropy numbers. Technical report, Departement of Statistics, University of Washington, 2002. URL `www.stat.washington.edu/www/research/reports/#2002/tr409.ps`.

Ingo Steinwart and Clint Scovel. Fast rates for support vector machines. In P. Auer and R. Meir, editors, *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pages 279–294, 2005.

Bernadetta Tarigan and Sara A. van de Geer. Classifiers of support machine type with l_1 complexity regularization. *Bernoulli*, 12(6):1045–1076, 2006.

Ambuj Tewari and Peter L. Bartlett. On the consistency of multiclass classification methods. In P. Auer and R. Meir, editors, *COLT*, volume 3559 of *Lecture Notes in Computer Science*, pages 143–157, 2005.

Alexandre B. Tsybakov. Optimal aggregation of classifiers in statistical learning. *Annals of Statistics*, 32:135–166, 2004.

Sara A. van de Geer. *Empirical Processes in M-estimation*. Cambridge University Press, 2000.

Aad W. van der Vaart and Jon A. Wellner. *Weak Convergence and Empirical Processes*. Springer Series in Statistics. Springer-Verlag, New York, 1996.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 2000.

Lifeng Wang and Xiaotong Shen. On l_1-norm multiclass support vector machines: Methodology and theory. *Journal of the American Statistical Association*, 102(478):583–594, 2007.

Jason Weston and Chris Watkins. Multi-class support vector machines. In *Proceedings of ESANN99*, 1999.

Tong Zhang. Statistical behaviour and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–134, 2004a. With discussion.

Tong Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004b.

Tong Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004c.

Hui Zou, Ji Zhu, and Trevor Hastie. The margin vector, admissible loss and multi-class margin-based classifiers. Technical report, Statistics Departement, Stanford University, 2006.

Journal of Machine Learning Research 9 (2008) 2187-2216 Submitted 4/07; Revised 6/08; Published 10/08

Ranking Individuals by Group Comparisons

Tzu-Kuo Huang R93002@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Ruby C. Weng CHWENG@NCCU.EDU.TW
Department of Statistics
National Chengchi University
Taipei 116, Taiwan

Editor: Greg Ridgeway

Abstract

This paper proposes new approaches to rank individuals from their group comparison results. Many real-world problems are of this type. For example, ranking players from team comparisons is important in some sports. In machine learning, a closely related application is classification using coding matrices. Group comparison results are usually in two types: binary indicator outcomes (wins/losses) or measured outcomes (scores). For each type of results, we propose new models for estimating individuals' abilities, and hence a ranking of individuals. The estimation is carried out by solving convex minimization problems, for which we develop easy and efficient solution procedures. Experiments on real bridge records and multi-class classification demonstrate the viability of the proposed models.

Keywords: ranking, group comparison, binary/scored outcomes, Bradley-Terry model, multi-class classification

1. Introduction

We address an interesting problem of estimating individuals' abilities from their group comparison results. This problem arises in some sports. One can evaluate a basketball player by his/her average points, but this criterion may be unfair as it ignores opponents' abilities. Comparison results in some sports, such as bridge, even do not reveal any direct information related to individuals' abilities. In a bridge match two partnerships form a team to compete with another two. The match record fairly reflects which two partnerships are better, but a partnership's raw score, depending on different boards, does not indicate its ability. Finding reasonable individual rankings using all group comparison records is thus a challenging task. Another application in machine learning/statistics is multi-class classification by coding matrices (Dietterich and Bakiri, 1995; Allwein et al., 2001). This technique decomposes a multi-class problem into several two-class problems, each of which is considered as the comparison between two disjoint subsets of class labels. The label with the greatest ability then serves as the prediction.

This line of research stems from the study of paired comparisons (David, 1988), in which one group/team consists of only one individual, and individuals' abilities are estimated from paired comparison results. Several models have been proposed, among which the most popular one is the

Bradley-Terry model (Bradley and Terry, 1952): suppose there are k individuals whose abilities are indicated by a non-negative vector $\mathbf{p} = [p_1\ p_2\ \ldots\ p_k]^T$. They proposed that

$$P(\text{individual } i \text{ beats } j) = \frac{p_i}{p_i + p_j}. \tag{1}$$

If comparisons are independent, then the maximum likelihood estimate of $\mathbf{p}$ is obtained by solving

$$\begin{aligned} \min_{\mathbf{p}} \quad & -\sum_{i \neq j} n_{ij} \log \frac{p_i}{p_i + p_j} \\ \text{subject to} \quad & \sum_{j=1}^{k} p_j = 1, \quad p_j \geq 0, j = 1, \ldots, k, \end{aligned} \tag{2}$$

where n_{ij} is the number of times individual i beats j. The normalizing constraint in (2) is imposed because the objective function is scale-invariant. The solution to (2) can be found via a simple iterative procedure, which converges to the unique global minimum under mild conditions. Detailed discussions are in, for example, Hunter (2004).

Going from paired to group comparisons, we consider k individuals $\{1, \ldots, k\}$ having m comparisons. The ith comparison setting involves a subset I_i, which is separated as two disjoint teams, I_i^+ and I_i^-. They have $n_i = n_i^+ + n_i^-$ comparisons, among which I_i^+ and I_i^- win n_i^+ and n_i^- times, respectively. Before seeking sophisticated models, an intuitive way to estimate the sth individual's ability is by the number of its winning comparisons normalized by the total number it involves:

$$\frac{\sum_{i:s \in I_i^+} n_i^+ + \sum_{i:s \in I_i^-} n_i^-}{\sum_{i:s \in I_i} n_i}. \tag{3}$$

In the case of paired comparisons, several authors (David, 1988; Hastie and Tibshirani, 1998) have shown that if

$$n_{si} > 0,\ n_{is} > 0, \text{ and } n_{si} + n_{is} = \text{constant}, \forall s, i, \tag{4}$$

then the ranking by (3) is identical to that by the solution of (2). Note that under (4), the denominator of (3) is the same over all s, so the calculation is simplified to

$$\sum_{i:i \neq s} n_{si},$$

Although the above property may provide some support of (3), this approach has several problems. Firstly, (4) may not hold in most applications of paired comparisons. Secondly, (3) does not consider teammates' abilities, so strong players and weak ones receive the same credits. Because of these deficiencies, we use (3) as a baseline in experiments in Section 4 to demonstrate the need for more advanced methods. We refer to this approach as AVG.

As a direct extension of (1), Huang et al. (2006b) proposed a generalized Bradley-Terry model for group comparisons:

$$P(I_i^+ \text{ beats } I_i^-) = \frac{\sum_{j:j \in I_i^+} p_j}{\sum_{j:j \in I_i} p_j}, \tag{5}$$

which assumes that a team's ability is the sum of its members'. Under the assumption that comparisons are independent, individuals' abilities can be estimated by minimizing the negative log-likelihood of (5):

$$\begin{aligned} \min_{\mathbf{p}} \quad & -\sum_{i=1}^{m} \left(n_i^+ \log \frac{\sum_{j:j\in I_i^+} p_j}{\sum_{j:j\in I_i} p_j} + n_i^- \log \frac{\sum_{j:j\in I_i^-} p_j}{\sum_{j:j\in I_i} p_j} \right) \\ \text{subject to} \quad & \sum_{j=1}^{k} p_j = 1, \quad p_j \geq 0, j = 1,\ldots,k. \end{aligned} \tag{6}$$

Huang et al. (2006b) pointed out that (6) may not be a convex optimization problem, so global minima are not easy to obtain. Zadrozny (2002) was the first attempt to solve (6) by an iterative procedure, which, however, may fail to converge to a stationary point (Huang et al., 2006b). The algorithm of Huang et al. (2006b) converges to a stationary point under certain conditions. We refer to this approach as GBT.ML (Generalized Bradley-Terry Model using Maximum Likelihood).

Both models (1) and (6) consider comparisons' "binary" outcomes, that is, wins and losses. However, in many comparisons, results are also quantities reflecting opponents' performances/strengths, such as points in basketball or soccer games. Some work use these "measured" outcomes for paired comparisons; an example is Glickman (1993): instead of modeling the probability that one individual beats another, he considers the difference in two individuals' abilities as a random variable, whose realization is the difference in two scores. Individuals' abilities are then estimated via maximizing the likelihood.

In this paper we focus on the *batch* setting, under which individuals' abilities are not estimated until all comparisons are finished. This setting is suitable for annual sports events, such as the Bermuda Bowl for bridge considered in Section 4, where the goal is to rank participants according to their performances in the event. However, in some applications, competitions continue to take place without a clear end and a real-time ranking is required. An example is online gaming, where players make teams to compete against one another anytime they wish and expect a real-time update of their ranking right after a game is over. Several work deal with such an *online* scenario. For example, Herbrich and Graepel (2007) proposed the TrueSkill$^{\text{TM}}$ system, which generalizes the Elo system used in Chess (Elo, 1986). The system follows a Bayesian framework and obtains real-time rankings by an online learning scheme called *Gaussian density filtering* (Minka, 2001). Menke and Martinez (2007) re-parameterized the Bradley-Terry model (2) as a single-layer artificial neural network (ANN) and extended it for group competitions. Individuals' abilities are estimated by training the ANN with the delta rule, a typical online or incremental learning technique.

We managed to advance the state of the art in two directions. On the one hand, for comparisons with binary outcomes, we propose a new exponential model in Section 2. The main advantage over Huang et al. (2006b) is that one can estimate individuals' abilities by minimizing unconstrained convex formulations. Hence global minima are easily obtained. On the other hand, we propose in Section 3 two models for comparisons with measured outcomes, which we call scored outcomes. The induced optimization problems are also unconstrained and convex; simple solution procedures are presented. This section may be the first study on finding individuals' abilities from *scored* group comparisons. Section 4 ranks partnerships in real bridge matches with the proposed approaches. Properties of different methods and their relations are studied in Section 5, which helps to explain experimental results. Section 6 demonstrates applications in multi-class classification. Section 7 concludes the work and discusses possible future directions.

Part of this work appears in a conference paper (Huang et al., 2006a).

2. Comparisons with Binary Outcomes

We denote individuals' abilities as a vector $\mathbf{v} \in R^k$, $-\infty < v_s < \infty$, $s = 1, \ldots, k$. Unlike $\mathbf{p}$ used in (5), $\mathbf{v}$ may have negative values. A team's ability is then defined as the sum of its members': for I_i^+ and I_i^-, their abilities are respectively

$$T_i^+ \equiv \sum_{s:s\in I_i^+} v_s \qquad \text{and} \qquad T_i^- \equiv \sum_{s:s\in I_i^-} v_s. \tag{7}$$

We consider teams' actual performances as random variables Y_i^+ and Y_i^-, $1 \le i \le m$ and define

$$P(I_i^+ \text{ beats } I_i^-) \equiv P(Y_i^+ - Y_i^- > 0). \tag{8}$$

The distribution of Y_i^+ and Y_i^- is generally unknown, but a reasonable choice should place the mode (the value at which the density function is maximized) around T_i^+ and T_i^-. To derive a computationally simple form for (8), we assume that Y_i^+ (and similarly Y_i^-) has a doubly-exponential extreme value distribution with

$$P(Y_i^+ \le y) = \exp(-e^{-(y-T_i^+)}), \tag{9}$$

whose mode is exactly T_i^+. Suppose Y_i^+ is independent of Y_i^-, from (8) and (9) we have

$$P(I_i^+ \text{ beats } I_i^-) = \frac{e^{T_i^+}}{e^{T_i^+} + e^{T_i^-}}. \tag{10}$$

The derivation is in Appendix A. One may assume other distributions (e.g., normal) in (9), but the resulting model is more complicated than (10). Such differences already occur for paired comparisons, where David (1988) gave some discussion. Thus (10) is our proposed model for binary outcomes.

For paired comparisons (i.e., each individual forms a team), (10) reduces to

$$P(\text{individual } i \text{ beats individual } j) = \frac{e^{v_i}}{e^{v_i} + e^{v_j}},$$

which is an equivalent re-parameterization (David, 1988; Hunter, 2004) of the Bradley-Terry model (1) by

$$p_i \equiv \frac{e^{v_i}}{\sum_{j=1}^k e^{v_j}}.$$

Therefore, our model (10) can also be considered as a generalized Bradley-Terry model. This re-parameterization however does not extend to the case of group comparisons, so (10) and (5) are different. Interestingly, (10) is a conditional exponential model or a *maximum entropy* model (Jaynes, 1957a,b), which is commonly used in the computational linguistic community (Berger et al., 1996). Thus we can use existing properties of this type of models.

Following the proposed model (10), we estimate $\mathbf{v}$ by using available comparison results. The following two sub-sections give two approaches: one minimizes a regularized least square formula, and the other minimizes the negative log-likelihood. Both are unconstrained convex optimization problems. Their differences are discussed in Section 5.

2.1 Regularized Least Square (Ext-B.RLS)

Recall that n_i^+ and n_i^- are respectively the number of comparisons teams I_i^+ and I_i^- win. From (10), we have

$$\frac{e^{T_i^+}}{e^{T_i^+}+e^{T_i^-}} \approx \frac{n_i^+}{n_i^+ + n_i^-},$$

and therefore

$$e^{T_i^+ - T_i^-} = \frac{e^{T_i^+}}{e^{T_i^-}} \approx \frac{n_i^+}{n_i^-}.$$

If $n_i^+ \neq 0$ and $n_i^- \neq 0$, one can solve

$$\min_{\mathbf{v}} \qquad \sum_{i=1}^{m} \left((T_i^+ - T_i^-) - \log \frac{n_i^+}{n_i^-} \right)^2 \tag{11}$$

to estimate the vector $\mathbf{v}$ of individuals' abilities. In case of $n_i^+ = 0$ or $n_i^- = 0$, a simple solution is adding a small number to all n_i^+ and n_i^-. This technique is widely used in the computational linguistic community, known as the "add-one smoothing" for dealing with the zero-frequency problem. To represent (11) in a simpler form, we define a vector $\mathbf{d} \in R^m$ with

$$d_i \equiv \log \frac{n_i^+}{n_i^-},$$

and a "comparison setting matrix" $G \in R^{m \times k}$ with

$$G_{ij} \equiv \begin{cases} 1 & \text{if individual } j \in I_i^+, \\ -1 & \text{if individual } j \in I_i^-, \\ 0 & \text{if individual } j \notin I_i. \end{cases} \tag{12}$$

Take bridge in teams of four as an example. An individual stands for a partnership, so G's jth column records the jth partnership's team memberships in all m matches. Since a match is played by four partnerships from two teams, each row of G has two 1's, two -1's and $k-4$ 0's. Thus, G may look like

$$\begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \tag{13}$$

read as "The first match: the 1st, 2nd partnerships versus the 3rd, 4th; the second match: the 1st, 2nd versus the 5th, 6th; ...."

With the help of $\mathbf{d}$ and G, we rewrite (11) as

$$\min_{\mathbf{v}} \qquad (G\mathbf{v} - \mathbf{d})^T (G\mathbf{v} - \mathbf{d}), \tag{14}$$

which is equivalent to solving the following linear system:

$$G^T G\mathbf{v} = G^T \mathbf{d}. \tag{15}$$

If $G^T G$ is not invertible, the linear system (15) may have multiple solutions, which lead to possibly multiple rankings. To see when $G^T G$ is invertible, we prove the following result:

Theorem 1 *G^TG is invertible if and only if rank$(G) = k$.*

The proof is in Appendix B. This result shows that teams' members should change frequently across comparisons (as indicated by *rank*$(G) = k$) so that individuals' abilities are uniquely determined. To see how multiple rankings occur, consider an extreme case where several players always belong to the same team. Under the model (10), they can be merged as a single virtual player. After solving (14), their respective abilities can take any values but still remain optimal as long as the total ability is equal to the virtual player's. To handle such situations, we add a regularization term $\mu\mathbf{v}^T\mathbf{v}$ to (14):

$$\min_{\mathbf{v}} \qquad (G\mathbf{v}-\mathbf{d})^T(G\mathbf{v}-\mathbf{d})+\mu\mathbf{v}^T\mathbf{v},$$

where μ is a small positive number. Then a unique solution exists:

$$\left(G^TG+\mu I\right)^{-1}G^T\mathbf{d}. \tag{16}$$

The rationale of the regularization is that individuals have equal abilities before having comparisons. We refer to this approach as Ext-B.RLS (Extreme value model for Binary outcomes using Regularized Least Square).

2.2 Maximum Likelihood (Ext-B.ML)

Under the assumption that comparisons are independent, the negative log-likelihood function is

$$l(\mathbf{v}) \equiv -\sum_{i=1}^{m}\left(n_i^+\log\frac{e^{T_i^+}}{e^{T_i^+}+e^{T_i^-}}+n_i^-\log\frac{e^{T_i^-}}{e^{T_i^+}+e^{T_i^-}}\right), \tag{17}$$

and we may estimate $\mathbf{v}$ by

$$\arg\min_{\mathbf{v}} l(\mathbf{v}).$$

It is well known that the log-likelihood of a conditional exponential model is concave, and hence $l(\mathbf{v})$ is convex. However, if $l(\mathbf{v})$ is not strictly convex, multiple global minima may result in multiple rankings. The following theorem gives the sufficient and necessary condition for strict convexity:

Theorem 2 *$l(\mathbf{v})$ is strictly convex if and only if rank$(G) = k$.*

The proof is in Appendix C. As discussed in Section 2.1, the condition may not hold, and a regularization term is usually added to ensure the uniqueness of the optimal solution. Here we consider a special one

$$\mu\sum_{s=1}^{k}(e^{v_s}+e^{-v_s}), \tag{18}$$

which is strictly convex and has unique minimum at $v_s = 0, s = 1,\ldots,k$. Later we will see that this function helps to derive a simple algorithm for maximizing the likelihood.

The modified negative log-likelihood is as the following:

$$l(\mathbf{v}) \equiv -\sum_{i=1}^{m}\left(n_i^+\log\frac{e^{T_i^+}}{e^{T_i^+}+e^{T_i^-}}+n_i^-\log\frac{e^{T_i^-}}{e^{T_i^+}+e^{T_i^-}}\right)+\mu\sum_{s=1}^{k}(e^{v_s}+e^{-v_s}), \tag{19}$$

where μ is a small positive number. We estimate individuals' abilities by the unique global minimum

$$\arg\min_{\mathbf{v}} l(\mathbf{v}), \tag{20}$$

which satisfies the optimality condition:

$$\begin{aligned}\frac{\partial l(\mathbf{v})}{\partial v_s} &= -\Big(\sum_{i:s\in I_i^+} n_i^+ + \sum_{i:s\in I_i^-} n_i^-\Big) + \sum_{i:s\in I_i^+} \frac{n_i e^{T_i^+}}{e^{T_i^+}+e^{T_i^-}} + \sum_{i:s\in I_i^-} \frac{n_i e^{T_i^-}}{e^{T_i^+}+e^{T_i^-}} + \mu(e^{v_s}-e^{-v_s}) \\ &= 0,\ \ s=1,\ldots,k.\end{aligned}$$

Note that the strict convexity of (19) may not guarantee (20) to be attainable; we address this issue later in Theorem 3. Since μ is small,

$$\sum_{i:s\in I_i^+} n_i^+ + \sum_{i:s\in I_i^-} n_i^- \approx \sum_{i:s\in I_i^+} \frac{n_i e^{T_i^+}}{e^{T_i^+}+e^{T_i^-}} + \sum_{i:s\in I_i^-} \frac{n_i e^{T_i^-}}{e^{T_i^+}+e^{T_i^-}}, \tag{21}$$

which is a reasonable condition that the total number of observed wins of individual s is nearly the expected number by the assumed model. Meanwhile, the last term in $\partial l(\mathbf{v})/\partial v_s$ restricts the value of v_s from extremity, and thereby brings some robustness against huge n_i^+ or n_i^-.

Standard optimization methods (e.g., gradient or Newton's method) can be used to find a solution of (19). For conditional exponential models, an alternative technique to maximize the likelihood is the generalized iterative scaling by Darroch and Ratcliff (1972), which generates a sequence of iterations $\{\mathbf{v}^t\}_{t=0}^{\infty}$. The improved iterative scaling (Pietra et al., 1997) speeds up the convergence, but its update from $\mathbf{v}^t$ to $\mathbf{v}^{t+1}$ requires the solution of k one-variable minimization problems, which, however, usually do not have closed-form solutions. Goodman (2002) proposed the sequential conditional generalized iterative scaling, which changes only one variable at a time with a closed-form update rule. All the above techniques, however, need to be modified for solving (19) due to the regularization term (18). In the following we propose an iterative method that modifies one component of $\mathbf{v}$ at a time. Let $\delta \equiv [0,\ldots,0,\delta_s,0,\ldots,0]^T$ indicate the change of the sth component. Using the inequality $\log x \le x-1,\ \forall x>0$,

$$\begin{aligned}& l(\mathbf{v}+\delta)-l(\mathbf{v}) \\ =& -\left(\sum_{i:s\in I_i^+} n_i^+ + \sum_{i:s\in I_i^-} n_i^-\right)\delta_s + \sum_{i:s\in I_i^+} n_i \log\left(\frac{e^{T_i^+}+e^{\delta_s}+e^{T_i^-}}{e^{T_i^+}+e^{T_i^-}}\right) \\ &+ \sum_{i:s\in I_i^-} n_i \log\left(\frac{e^{T_i^+}+e^{T_i^-+\delta_s}}{e^{T_i^+}+e^{T_i^-}}\right) + \mu e^{v_s}(e^{\delta_s}-1) + \mu e^{-v_s}(e^{-\delta_s}-1) \\ \le& -\left(\sum_{i:s\in I_i^+} n_i^+ + \sum_{i:s\in I_i^-} n_i^-\right)\delta_s + \left(\sum_{i:s\in I_i^+} \frac{n_i e^{T_i^+}}{e^{T_i^+}+e^{T_i^-}} + \sum_{i:s\in I_i^-} \frac{n_i e^{T_i^-}}{e^{T_i^+}+e^{T_i^-}}\right)(e^{\delta_s}-1) \\ &+ \mu e^{v_s}(e^{\delta_s}-1) + \mu e^{-v_s}(e^{-\delta_s}-1).\end{aligned} \tag{22}$$

If $\delta_s = 0$, (22) $= 0$. We then minimize (22) to obtain the largest reduction. It is easy to see that (22) is strictly convex. Taking the derivative with respect to δ_s to be zero, we find the root for a

second-order polynomial of e^{δ_s}, so the update rule is:

$$v_s \leftarrow v_s + \log \frac{B_s + \sqrt{B_s^2 + 4\mu A_s e^{-v_s}}}{2A_s}, \tag{23}$$

where

$$\begin{aligned} A_s &\equiv \mu e^{v_s} + \sum_{i:s\in I_i^+} \frac{n_i e^{T_i^+}}{e^{T_i^+} + e^{T_i^-}} + \sum_{i:s\in I_i^-} \frac{n_i e^{T_i^-}}{e^{T_i^+} + e^{T_i^-}}, \\ B_s &\equiv \sum_{i:s\in I_i^+} n_i^+ + \sum_{i:s\in I_i^-} n_i^-. \end{aligned} \tag{24}$$

If using other regularization terms, minimizing (22) may not lead to a closed-form solution of δ_s. The algorithm is as the following:

Algorithm 1

1. Start with $\mathbf{v}^0$ and obtain $T_i^{0,+}, T_i^{0,-}, i = 1,\ldots,m$.
2. Repeat ($t = 0, 1, \ldots$)
   (a) Let $s = (t+1) \mod k$. Change the sth element of $\mathbf{v}^t$ by (23) to obtain $\mathbf{v}^{t+1}$.
   (b) Calculate $T_i^{t+1,+}, T_i^{t+1,-}, i = 1,\ldots,m$.

   until $\partial l(\mathbf{v}^t)/\partial v_j = 0,\ j = 1,\ldots,k$ are satisfied.

Next we address the convergence issue. As $A_s > 0$, (23) is always well-defined. A formal proof of Algorithm 1's convergence is in the following theorem:

Theorem 3 *The modified negative log-likelihood $l(\mathbf{v})$ defined in* (19) *attains a unique global minimum, and the sequence $\{\mathbf{v}^t\}$ generated by Algorithm 1 converges to it.*

The proof is in Appendix D. In Huang et al. (2006b), some assumptions are needed to ensure their update rule to be well-defined as well as the convergence. In contrast, Algorithm 1 does not require any assumption since the regularization term provides very nice properties. We refer to the approach of minimizing (19) as Ext-B.ML (Extreme value distribution model for Binary outcomes using Maximum Likelihood).

3. Comparisons with Scored Outcomes

This section proposes estimating individuals' abilities based on measured outcomes, such as points in basketball or soccer games. We still use random variables Y_i^+ and Y_i^- for team performances, but give n_i^+ and n_i^- different meanings: they now denote scores of I_i^+ and I_i^-. Our idea is to view $n_i^+ - n_i^-$ as a realization of $Y_i^+ - Y_i^-$ and maximize the resulting likelihood. Note that we model *difference* in scores instead of the score itself. We propose two approaches in the following subsections. One assumes normal distributions for Y_i^+ and Y_i^-, while the other assumes the same extreme value distribution (9). Individuals' abilities are estimated by maximizing the likelihood of score differences. Properties of the two approaches are investigated in Section 5.

3.1 Normal Distribution Model (NM-S.ML)

As mentioned in Section 2, using normal distributions for comparisons with binary outcomes is computationally more difficult due to a complicated form of $P(I_i^+ \text{ beats } I_i^-)$. However, for scored paired comparisons, Glickman (1993) successfully applied normal distributions. He considers individuals' performances as normally distributed random variables

$$Y_i \sim N(v_i, \sigma^2),\ i = 1, \ldots, k,$$

and view the score difference of individuals i and j as a realization of $Y_i - Y_j$. By assuming Y_i and Y_j are independent for all individuals,

$$Y_i - Y_j \sim N(v_i - v_j, 2\sigma^2), \tag{25}$$

and individuals' abilities are estimated by maximizing the likelihood. We extend this approach to group comparisons. Recall that Y_i^+ and Y_i^- are random variables for two teams' performances. With the same assumption of independent normal distributions, we have

$$Y_i^+ \sim N(T_i^+, \sigma^2), \quad Y_i^- \sim N(T_i^-, \sigma^2).$$

and

$$Y_i^+ - Y_i^- \sim N(T_i^+ - T_i^-, 2\sigma^2).$$

Assuming comparisons are independent and defining a vector $\mathbf{b}$ with

$$b_i \equiv n_i^+ - n_i^-,$$

the negative log-likelihood then is

$$\begin{aligned} l(\mathbf{v}, \sigma) &= \log\sigma + \frac{1}{4\sigma^2} \sum_{i=1}^{m} \left(T_i^+ - T_i^- - (n_i^+ - n_i^-)\right)^2 \\ &= \log\sigma + \frac{(G\mathbf{v} - \mathbf{b})^T (G\mathbf{v} - \mathbf{b})}{4\sigma^2}, \end{aligned} \tag{26}$$

where G is the comparison setting matrix defined in (12). The maximum likelihood estimate of $\mathbf{v}$ is obtained by solving $\partial l(\mathbf{v}, \sigma)/\partial v_s = 0\ \forall s$, which is the following linear system:

$$G^T G\mathbf{v} = G^T \mathbf{b}. \tag{27}$$

Similar to (14), (27) may have multiple solutions if $G^T G$ is not invertible. To overcome this problem, we add a regularization term and solve

$$\min_{\mathbf{v}} \quad l(\mathbf{v}, \sigma) + \frac{\mu}{4\sigma^2} \mathbf{v}^T \mathbf{v}, \tag{28}$$

where μ is small positive number. The unique solution of (28) then is

$$\bar{\mathbf{v}} \equiv (G^T G + \mu I)^{-1} G^T \mathbf{b}. \tag{29}$$

In addition, we also obtain an estimate of the variance by solving

$$\frac{\partial \left(l(\mathbf{v}, \sigma^2) + \frac{\mu}{4\sigma^2} \mathbf{v}^T \mathbf{v}\right)}{\partial \sigma} = 0,$$

which leads to

$$\bar{\sigma}^2 \equiv \frac{(G\bar{\mathbf{v}} - \mathbf{b})^T (G\bar{\mathbf{v}} - \mathbf{b}) + \mu \bar{\mathbf{v}}^T \bar{\mathbf{v}}}{2}.$$

We refer to this method as NM-S.ML (Normal distribution-based Model for Scored outcomes using Maximum Likelihood).

3.2 Extreme Value Distribution Model (Ext-S.ML)

Instead of the normal distribution in (25), we now consider that $Y_i^+ - Y_i^-$ is under the extreme value distribution for binary outcomes. Appendix A shows that

$$P(Y_i^+ - Y_i^- \le y) = \frac{e^{T_i^-}}{e^{T_i^+ - y} + e^{T_i^-}}, \tag{30}$$

and hence the density function is

$$f_{Y_i^+ - Y_i^-}(y) = \frac{e^{T_i^- + T_i^+ - y}}{(e^{T_i^+ - y} + e^{T_i^-})^2}.$$

The negative log-likelihood function is

$$-\sum_{i=1}^{m} \log \frac{e^{T_i^+ + T_i^- - (n_i^+ - n_i^-)}}{\left(e^{T_i^+ - (n_i^+ - n_i^-)} + e^{T_i^-}\right)^2}. \tag{31}$$

A similar proof to Theorem 2's shows that (31) is convex and shares the same condition for strict convexity in Section 2.2. Therefore, the problem of multiple solutions may also occur. We thus adopt the same regularization term as in Section 2.2 and solve

$$\min_{\mathbf{v}} \quad l(\mathbf{v}) \equiv -\sum_{i=1}^{m} \log \frac{e^{T_i^+ + T_i^- - (n_i^+ - n_i^-)}}{\left(e^{T_i^+ - (n_i^+ - n_i^-)} + e^{T_i^-}\right)^2} + \mu \sum_{s=1}^{k} (e^{v_s} + e^{-v_s}). \tag{32}$$

The unique global minimum satisfies for $s = 1, \ldots, k$,

$$\begin{aligned} \frac{\partial l(\mathbf{v})}{\partial v_s} &= -m_s + 2\left(\sum_{i:s \in I_i^+} \frac{e^{T_i^+ + n_i^-}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} + \sum_{i:s \in I_i^-} \frac{e^{T_i^- + n_i^+}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} \right) + \mu(e^{v_s} - e^{-v_s}) \\ &= 0, \end{aligned} \tag{33}$$

where

$$m_s \equiv \sum_{i:s \in I_i} 1.$$

From (30),

$$P(Y_i^+ - Y_i^- \ge T_i^+ - T_i^-) = \frac{1}{2},\ i = 1, \ldots, m. \tag{34}$$

Since μ is small, (33) and (34) imply that for $s = 1, \ldots, k$,

$$\begin{aligned} &\sum_{i:s \in I_i^+} P(Y_i^+ - Y_i^- \ge n_i^+ - n_i^-) + \sum_{i:s \in I_i^-} P(Y_i^- - Y_i^+ \ge n_i^- - n_i^+) \\ \approx \frac{m}{2} &= \sum_{i:s \in I_i^+} P(Y_i^+ - Y_i^- \ge T_i^+ - T_i^-) + \sum_{i:s \in I_i^-} P(Y_i^- - Y_i^+ \ge T_i^- - T_i^+). \end{aligned}$$

As (21) in Section 2.2, the above condition also indicates that models should be consistent with observations. To solve (32), we use Algorithm 1 with a different update rule, which is in the form of (23) but with

$$\begin{aligned} A_s &\equiv \mu e^{v_s} + 2\left(\sum_{i:s \in I_i^+} \frac{e^{T_i^+ + n_i^-}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} + \sum_{i:s \in I_i^-} \frac{e^{T_i^- + n_i^+}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} \right), \\ B_s &\equiv m_s. \end{aligned}$$

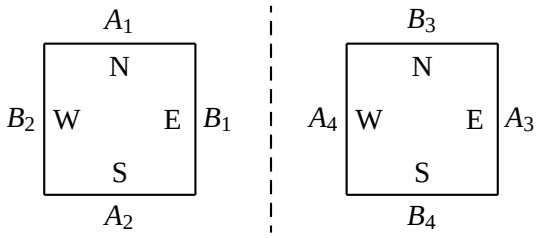


Figure 1: A typical bridge match setting. N, S, E and W stand for north, south, east, and west, respectively.

The derivation is similar to (23)'s: let $\delta \equiv [0,\ldots,0,\delta_s,0,\ldots,0]^T$. Then

$$\begin{aligned}
& l(\mathbf{v}+\delta) - l(\mathbf{v}) \\
=& -m_s\delta_s + 2\left(\sum_{i:s\in I_i^+} \log \frac{e^{T_i^+ + n_i^- + \delta_s} + e^{T_i^- + n_i^+}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} + \sum_{i:s\in I_i^-} \log \frac{e^{T_i^- + n_i^+ + \delta_s} + e^{T_i^+ + n_i^-}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} \right) \\
& + \mu e^{v_s}(e^{\delta_s} - 1) + \mu e^{-v_s}(e^{-\delta_s} - 1) \\
\leq& -m_s\delta_s + 2\left(\sum_{i:s\in I_i^+} \frac{e^{T_i^+ + n_i^-}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} + \sum_{i:s\in I_i^-} \frac{e^{T_i^- + n_i^+}}{e^{T_i^+ + n_i^-} + e^{T_i^- + n_i^+}} \right)(e^{\delta_s} - 1) \\
& + \mu e^{v_s}(e^{\delta_s} - 1) + \mu e^{-v_s}(e^{-\delta_s} - 1). \qquad (35)
\end{aligned}$$

Minimizing (35) leads to the update rule. Global convergence can be proved in a similar way to Theorem 3. We refer to this approach as Ext-S.ML (Extreme value distribution model for Scored outcomes using Maximum Likelihood).

4. Ranking Partnerships from Real Bridge Records

This section presents a real application: ranking partnerships from match records of Bermuda Bowl 2005,[1] which is the most prestigious bridge event. In a match two partnerships (four players) from a team compete with two from another team. The rules require mutual understanding within a partnership, so partnerships are typically fixed while a team can send different partnerships for different matches. To rank partnerships using our model, an individual stands for a partnership, and every T_i^+ (or T_i^-) consists of two individuals. We caution the use of the term "team" here. Earlier we refer to each T_i^+ as a team and in bridge the two partnerships (or four players) of T_i^+ are really called a team. However, these four players are from a (super)-team (usually a country), which often has six members. We use "team" in both situations, which are easily distinguishable.

4.1 Experimental Settings

We discuss why a partnership's ability is not directly available from match results, and explain why our model is applicable here. Figure 1 illustrates the match setting. A_1, A_2, A_3, A_4 and B_1, B_2, B_3, B_4

1. All match records are available at `http://www.worldbridge.org/tourn/Estoril.05/Estoril.htm`. The subset used here is available at `http://www.csie.ntu.edu.tw/~cjlin/papers/genBTexp/Data.zip`.

Board	Table I		Table II		IMPs	
	NS	EW	NS	EW	IN	PT
1		1510		1510		
2	100		650			11
3		630		630		
4		650		660		
5	690		690			
6	420			50	10	
7	140		600			10
8		420		100		8
9	460		400		2	
10		110		140	1	

Table 1: Records of the first ten boards between India (IN) and Portugal (PT). India: NS at Table I and EW at Table II. The four columns in the middle are boards' raw scores, and only winners get points. For example, in the second board IN's NS partnership won at Table I and got 100 points while PT's NS got 650 at Table II. Since PT got more points than IN, it obtained IMPs.

are four players of Team *A* and Team *B*, sitting at two tables as depicted. A match consists of several boards, each of which is played at both tables. An important feature is that a board's four hands are at identical positions of two tables, but a team's two partnerships sit at complementary positions. In Figure 1, A_1 and A_2 sit at the north (N) and the south (S) sides of one table, so A_3 and A_4 must sit at the east (E) and the west (W) sides of the other table. This setting reduces the effect of uneven hands.

On each board winning partnerships receive raw scores. Depending on the difference in two teams' total scores, the winning team gains International Match Points (IMPs). For example, Table 1 shows records of the first ten boards of the match between two Indian partnerships and two Portuguese partnerships. We can see that a larger difference in raw scores results in more IMPs for the winner. IMPs are then converted to Victory Points (VP) for the team ranking.[2] A quick look at Table 1 may motivate the following straightforward approach: a partnership's score in a match is the sum of raw scores over all boards, and its ability is the average over the matches it plays. However, this estimate is unfair due to raw scores' dependency on boards and opponents. Summing a partnership's raw scores favors those who get better hands or play against weak opponents. Moreover, since boards are different across rounds and partnerships play in different rounds, the sum of raw scores can be more unfair. The above analysis indicates that a partnership's ability cannot be obtained directly from group comparison results. Hence the proposed models can be helpful.

We consider qualifying games: 22 teams from all over the world had a round robin tournament, which consisted of $\binom{22}{2} = 231$ matches and each team played 21. Most teams had six players in three fixed partnerships, and there were 69 partnerships in total. In order to obtain reasonable rankings, each partnership should play enough matches. The last column of Table 3 shows each partnership's

2. The IMP-to-VP conversion for Bermuda Bowl 2005 is on page 32, http://www.worldbridge.org/departments/rules/GeneralConditionsOfContest2005.pdf.

number of matches. Most played 13 to 15 matches, which are close to the average (14=21×2/3) of a team with three fixed partnerships. Thus these match records are reasonable for further analysis.

To use our model, the comparison setting matrix G defined in (12) is of size 231×69; as shown in (13) each row records a match setting and has exactly two 1's (two partnerships from one team), two -1's (two partnerships from another team) and 65 0's (the remaining partnerships). The sum of two rival teams' scores (VPs) is generally 30, but occasionally between 25 to 30 as a team's maximal VP is 25. We use two rival teams' VPs as n_i^+ and n_i^-, respectively. Several matches have zero scores; we add one to all n_i^+ and n_i^- for Ext-B.RLS to avoid the numerical difficulties caused by $\log(n_i^+/n_i^-)$.

4.2 Evaluation and Results

In sport events, rankings serve two main purposes. On the one hand, they summarize the relative performances of players or teams based on outcomes in the event, so that people may easily distinguish outstanding ones from poor ones. On the other hand, rankings in past events may indicate the outcomes of future events, and can therefore become a basis for designing future event schedules. Interestingly, we may connect these two purposes with two basic concepts in machine learning: minimizing the empirical error and minimizing the generalization error. For the first purpose, a good ranking must be consistent with available outcomes of the event, which relates to minimizing errors on training data, while for the second, a good ranking must predict well on the outcomes of future events, which is about minimizing errors on unseen data. We thus adopt these two principles to evaluate the proposed approaches, and in the context of bridge matches, they translate into the following evaluation criteria:

- **Empirical Performance**: How well do the estimated abilities and rankings fit the available match records?
- **Generalization Performance**: How well do the estimated abilities and rankings predict the outcomes of unseen matches?

Here we distinguish individuals' abilities from their ranking: Abilities give a ranking, but not vice versa. When we only have a ranking of individuals, groups' strengths are not directly available since the relation of individuals' ranks to those of groups is unclear. In contrast, if individuals' abilities are available, a group's ability can be the sum of its members'. We thus propose different error measures for abilities and rankings. Let $\{(I_1^+, I_1^-, n_1^+, n_1^-), \ldots, (I_m^+, I_m^-, n_m^+, n_m^-)\}$ be the group comparisons of interest and their outcomes. For a vector $\mathbf{v} \in R^k$ of individuals' abilities, we define the

- **Group Comparison Error**:

$$GCE(\mathbf{v}) \quad \equiv \quad \frac{\sum_{i=1}^{m} I\left\{(n_i^+ - n_i^-)(T_i^+ - T_i^-) \leq 0\right\}}{m},$$

where $I\{\cdot\}$ is the indicator function; T_i^+ and T_i^- are predicted group abilities of I_i^+ and I_i^-, as defined in (7). The GCE is essentially the proportion of wrongly predicted comparisons by the ability vector $\mathbf{v}$ to the m comparisons.

In the error measure for rankings, we use $\mathbf{r}$, a permutation of the k individuals, to denote a ranking, where r_s is the rank of individual s. Then we define the

- **Group Rank Error**:

$$GRE(\mathbf{r}) \equiv \frac{\sum_{i=1}^{m}\left(I\left\{n_i^+ > n_i^- \text{ and } U_i^+ > L_i^-\right\} + I\left\{n_i^+ < n_i^- \text{ and } L_i^+ < U_i^-\right\}\right)}{\sum_{i=1}^{m}\left(I\left\{U_i^+ > L_i^-\right\} + I\left\{L_i^+ < U_i^-\right\}\right)}, \quad (36)$$

in which

$$U_i^+ \equiv \min_{j\in I_i^+} r_j, \qquad L_i^+ \equiv \max_{j\in I_i^+} r_j,$$
$$U_i^- \equiv \min_{j\in I_i^-} r_j, \qquad L_i^- \equiv \max_{j\in I_i^-} r_j.$$

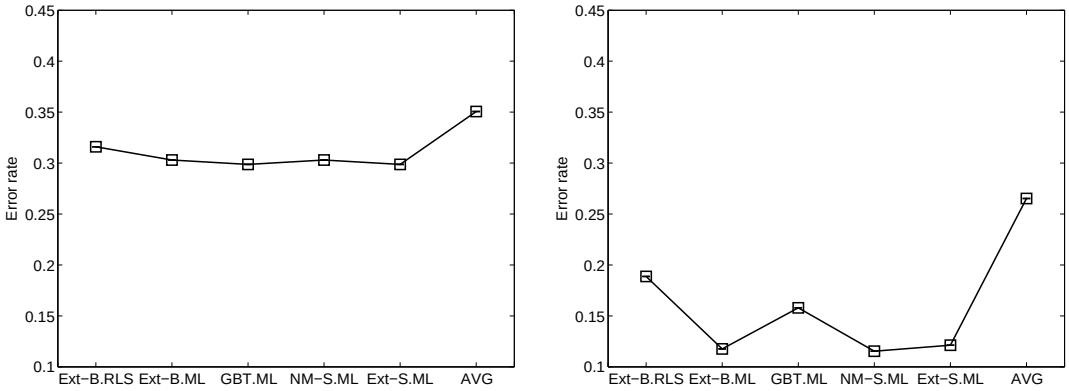
Since a smaller rank indicates more strength, the U_i^+ and L_i^+ defined above represent the best and the weakest in I_i^+, respectively. The denominator in (36) is thus the number of comparisons where one group's members are all ranked higher (or lower) than the members of the competing group, and the numerator in (36) counts the number of wrong predictions, that is, comparisons in which members of the winning group are all ranked lower than those of the defeated group. In other words, GRE computes the error only on comparisons in which relative strengths of the participating groups can be clearly determined by their members' ranks, whereas GCE considers the error on all of the comparisons. From this point of view, GRE is a more conservative error measure.

Combining the two error measures with the two evaluation criteria, we conducted four sets of experiments: Empirical GCE, Empirical GRE, Generalization GCE, and Generalization GRE. We compared six approaches, including the newly proposed Ext-B.RLS, Ext-B.ML, NM-S.ML, and Ext-S.ML; the generalized Bradley-Terry model GBT.ML (Huang et al., 2006b), and AVG, the simple approach (3) of summing individuals' scores, which serves as a baseline.[3] In the empirical part, we applied each approach on the entire 231 matches to estimate partnerships' abilities, and computed the two errors. Since the goal in the empirical part is to fit available records well, we set the regularization parameter μ for all approaches[4] except AVG to a small value 10^{-3}. In the generalization part, we randomly split the entire set as a training set of 162 matches and a testing set of 69 matches for 50 times. For each split, we searched for μ in $[2^5, 2^4, \ldots, 2^{-8}, 2^{-9}]$ by the Leave-One-Out (LOO) validation on the training set, estimated partnerships' abilities with the best μ, and then computed GCE and GRE on the testing set.

Results are in Figures 2 and 3 for empirical and generalization performances, respectively. In the empirical part, the four proposed approaches and GBT.ML perform obviously better than AVG, and the improvement in GRE is very significant. In particular, Ext-B.ML, NM-S.ML, and Ext-S.ML cause very small GREs, to the order of 10^{-1}. These results show that the proposed approaches are effective in fitting the available bridge match records. However, in the generalization part, all of the approaches result in poor GCEs, nearly as large as a random predictor does, and the proposed approaches did not improve over AVG. For GREs, values are smaller, but the improvements over

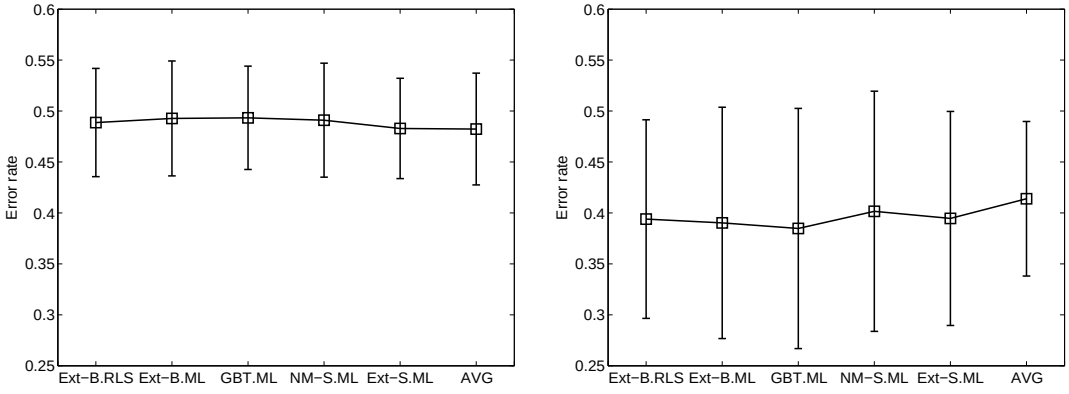
3. AVG gives individuals' abilities. We then use the same summation assumption to obtain groups' abilities for computing GCEs.

4. In order to ensure the convergence of their algorithm, Huang et al. (2006b) added to the objective function (5) what they called a "barrier term," which is also controlled by a small positive number μ (See Eq. (14) in Huang et al. 2006b). Here we simply refer to it as a regularization parameter.



(a) Group Comparison Error (b) Group Rank Error

Figure 2: Empirical performances of the six approaches.



(a) Group Comparison Error (b) Group Rank Error

Figure 3: Generalization performances of the six approaches, averaged over 50 random testing sets. Vertical bars indicate standard deviations.

Ext-B.RLS	Ext-B.ML	GBT.ML	NM-S.ML	Ext-S.ML	AVG
10/53	6/51	9/57	6/52	8/66	35/132

Table 2: Empirical Group Rank Errors in fraction.

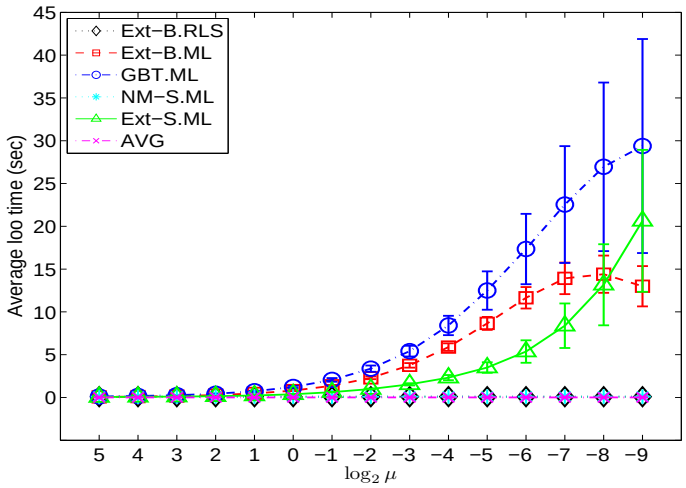


Figure 4: Average LOO time (sec) over 50 training/testing splits. Vertical bars indicate standard deviations.

AVG are rather marginal. In the following we give some accounts of the poor generalization performances. As mentioned in Section 4.1, each match setting can be viewed as a vector in $\{1,0,-1\}^{69}$, in which only two dimensions have 1's, and another two have -1's. Moreover, we are using records in the qualifying stage, a round-robin tournament in which every two teams (countries) played exactly one match. Consequently, when a match is removed from the training set, the four competing partnerships of that match have no chance to meet directly during the training stage. Indirect comparisons may only be marginally useful in predicting those partnerships' competition outcomes due to the lack of transitivity. In conclusion, the outcome of a match in this bridge data set may not be well indicated by outcomes of the other matches, and therefore all of the approaches failed to generalize well.

To further study the rankings by the six approaches, we show in Table 2 their empirical GREs. Since GRE only looks at the subset of matches in which group members' ranks clearly decide groups' relative strengths, the size of this subset, that is, the denominator in GRE, may also be a performance indicator of each approach. We thus present GREs in fraction. It is clear that the ranking by AVG is able to determine the outcomes of more matches, but at the same time causes more errors. Similar results are also found in the generalization experiments. We may therefore say that the proposed approaches and GBT.ML lead to rankings with more "precision," in the sense that they may not be able to decide groups' relative performances in the majority of comparisons, but once they do, their decisions are accurate.

In addition to the efficacy of the six approaches, we also reported their efficiency. Figure 4 shows the average LOO time over the 50 training/testing splits under different values of μ. We obtained these timing results on an Intel® Core™2 Quad CPU (2.66GHz) machine with 8G main

memory; the linear systems of Ext-B.RLS and NM-S.ML were solved by Gaussian Elimination. AVG,[5] Ext-B.RLS, and NM-S.ML finished LOO almost instantly under all values of μ, while Ext-B.ML, GBT.ML, and Ext-S.ML, the three approaches using iterative algorithms, took more time as μ decreased. However, for large-scale problems with a huge k or m, traditional linear system solvers may encounter memory or computational difficulties, and the efficiency of the proposed approaches requires a more thorough study.

Finally, we list the top ten partnerships ranked by Ext-B.ML in Appendix F. Most of them are famous bridge players.

5. Properties of Different Approaches

Although we distinguish binary comparisons from scored ones, they are similar in some situations. On the one hand, if two teams had a series of comparisons, the number of victories can be viewed as a team's score in a super-game. On the other hand, scores in a game might be the sum of binary outcomes; for example, scores in soccer games are total numbers of successful shots. It is therefore interesting to study the properties of different methods and their relation. Table 3 lists partnership rankings obtained by applying the six approaches to the entire set of match records. We first investigate the similarity between these rankings by Kendall's tau, a standard correlation coefficient that quantifies the consistency between two rankings. We computed Kendall's tau for every pair of the six rankings and present them in Table 4, which indicates roughly three groups: Ext-B.RLS, Ext-B.ML, GBT.ML and NM-S.ML give similar rankings; the one by Ext-S.ML is quite different, while AVG seems to be uncorrelated with the others. We then measure the distance between two groups of rankings $g1$ and $g2$: For each partnership,

$$\begin{aligned} & d(\text{ranks by } g1, \text{ranks by } g2) \\ \equiv & \begin{cases} \min(\text{ranks by } g2) - \max(\text{ranks by } g1) & \text{if ranks by } g1 \text{ are all smaller,} \\ \min(\text{ranks by } g1) - \max(\text{ranks by } g2) & \text{if ranks by } g2 \text{ are all smaller,} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{37}$$

For example, from Table 3 the second partnership of U.S.A.2 is ranked 67th/65th/67th/65th by Ext-B.RLS/Ext-B.ML/GBT.ML/NM-S.ML and 25th by AVG. Therefore,

$$d(\{67,65,67,65\},25) = \min(67,65,67,65) - 25 = 40.$$

Checking all 69 partnerships' ranks gives

$$|d(\{\text{Ext-B.RLS,Ext-B.ML,GBT.ML,NM-S.ML}\}, \text{Ext-S.ML}) \geq 20| = 6, \tag{38}$$

$$|d(\{\text{Ext-B.RLS,Ext-B.ML,GBT.ML,NM-S.ML}\}, \text{AVG}) \geq 20| = 11. \tag{39}$$

In Table 3 we respectively underline and boldface partnerships satisfying (38) and (39). The eleven ranks satisfying (39) shows that AVG's ranking is closer to the team ranking:[6] Partnerships satisfying (39) have higher ranks than those by the others when the team ranks are high, but have the opposite when the team ranks are low. This observation indicates that AVG may fail to identify weak (strong) individuals from strong (weak) groups.

5. Apparently there is no need to run LOO for AVG, which is independent of μ; we do it here only for timing comparisons.

6. Recall that in the beginning of Section 4, we mentioned that all teams, after the qualifying stage was over, were ranked according to their total VPs gained in the tournament.

Team (ordered by team rankings)	Partnership rankings																		#match		
	Ext-B.RLS			Ext-B.ML			GBT.ML			NM-S.ML			Ext-S.ML			AVG					
Italy (IT)	14	18	11	7	14	21	4	12	19	6	18	22	7	4	40	5	4	11	15	14	13
U.S.A.2 (US2)	57	**67**	1	53	**65**	1	39	**67**	1	53	**65**	1	54	50	1	42	**25**	2	8	17	17
U.S.A.1 (US1)	8	27	**37**	11	17	**38**	11	13	**38**	11	14	**38**	35	9	16	23	6	**10**	18	10	14
Sweden (SE)	2	43	50	2	23	55	2	10	65	2	23	56	5	8	47	1	14	38	14	13	15
India (IN)	10	35	**39**	9	29	**41**	9	28	**40**	9	28	**41**	12	28	37	19	12	**15**	15	14	13
Argentina (AR)	29	25	28	27	20	30	25	23	34	26	19	30	41	10	52	16	18	26	15	14	13
Egypt (EG)	47	23	24	51	18	22	51	22	15	51	17	21	51	18	17	37	20	3	14	20	7
	49			**52**			**50**			**52**			44			**8**			1		
Brazil (BR)	31	4	**66**	28	8	**59**	24	8	**63**	29	7	**58**	26	57	11	28	13	**31**	11	18	13
Japan (JP)	5	**65**	38	3	**67**	39	3	**68**	27	3	**68**	40	3	68	15	7	**44**	46	14	14	14
Netherlands (NL)	16	52	17	32	43	31	30	45	33	32	43	31	30	34	49	36	32	24	15	15	12
China (CN)	51	48	7	45	44	6	47	46	7	45	44	8	43	31	21	30	52	9	13	14	15
South Africa (ZA)	45	30	20	49	26	15	52	26	20	50	24	15	55	29	13	49	35	27	15	13	14
Russia (RU)	34	21	42	35	16	46	36	16	49	36	16	47	48	6	23	39	21	53	14	14	14
Portugal (PT)	22	12	58	34	10	56	29	14	60	37	10	55	33	22	56	50	29	47	14	14	14
Australia (AU)	40	55	19	42	50	19	43	53	21	42	49	20	20	45	32	43	51	40	16	11	15
New Zealand (NZ)	68	41	3	68	48	5	66	42	5	66	48	5	66	58	2	64	41	17	9	16	17
England (UK)	**9**	33	61	**12**	36	64	**17**	32	64	**13**	35	63	36	25	64	**48**	22	55	17	12	13
Canada (CA)	13	36	56	13	40	58	18	35	62	12	39	57	19	24	67	34	45	62	14	16	12
Chinese Taipei (TW)	53	62	46	63	66	57	56	61	55	64	67	59	59	65	60	57	56	66	2	12	1
	6	**26**	59	**4**	**25**	54	**6**	**37**	54	**4**	**25**	54	14	27	53	**33**	**63**	61	4	7	16
Poland (PL)	**15**	54	60	**24**	47	60	**31**	48	59	**27**	46	60	39	38	61	**58**	54	60	12	15	15
Guadeloupe (GP)	**44**	32	69	**37**	33	69	**44**	41	69	**34**	33	69	42	46	69	**65**	59	69	14	14	14
Jordan (JO)	63	64		62	61		57	58		61	62		62	63		67	68		21	21	

Table 3: Partnerships' rankings. A partnership corresponds to the same position in columns. For example, Italy's second partnership is ranked 18th, 14th, 12th, 18th, 4th and 4th by Ext-B.RLS, Ext-B.ML, GBT.ML, NM-S.ML, Ext-S.ML and AVG, respectively, and it plays 14 matches. Rankings satisfying (38) and (39) are underlined and boldfaced, respectively.

	Ext-B.RLS	Ext-B.ML	GBT.ML	NM-S.ML	Ext-S.ML	AVG
Ext-B.RLS	1.00	0.84	0.79	0.82	0.50	0.44
Ext-B.ML	0.84	1.00	0.87	0.97	0.61	0.49
GBT.ML	0.79	0.87	1.00	0.86	0.62	0.53
NM-S.ML	0.82	0.97	0.86	1.00	0.60	0.49
Ext-S.ML	0.50	0.61	0.62	0.60	1.00	0.50
AVG	0.44	0.49	0.53	0.49	0.50	1.00

Table 4: Kendall's tau (correlation coefficients).

The above results suggest that approaches based on different types of comparisons may produce similar rankings, such as Ext-B.ML and NM-S.ML, while those based on the same type of outcomes may lead to diverse results, such as NM-S.ML and Ext-S.ML. Therefore, in the next two subsections we study their formulations and obtain the following results:

- When all n_i's are equal, that is, the number of games or the total score in every group comparison is the same, and estimated group abilities are approximately even, Ext-B.ML and NM-S.ML give similar rankings.
- When all n_i's are equal, Ext-B.RLS is more sensitive than Ext-B.ML and NM-S.ML to extreme outcomes ($n_i^+ \approx 0$ or $n_i^+ \approx n_i$).
- For the two scored-outcome approaches, extreme outcomes have a greater impact on NM-S.ML than on Ext-S.ML.

5.1 Comparing Binary- and Scored-outcome Approaches

Experimental results in Section 4 show that the binary-outcome approach Ext-B.ML and the scored-outcome approach NM-S.ML give very similar rankings. By analyzing their optimization problems, we find that

Claim 1 *If all n_i's are equal and the optimal* **v** *for Ext-B.ML satisfies*

$$T_i^+ \approx T_i^- \quad \forall i,$$

then Ext-B.ML and NM-S.ML give very close rankings.

The proof is in Appendix E. For the bridge data used in Section 4, n_i's are two rival teams' total VPs and are mostly 30; the average $|T_i^+ - T_i^-|$ from the optimal **v** for Ext-B.ML is 0.3983.

However, in applications where n_i's are unequal, these two approaches may give different results. Clearly, they use different approximations:

$$\frac{e^{T_i^+}}{e^{T_i^-}} \approx \frac{n_i^+}{n_i^-} \qquad \text{and} \qquad T_i^+ - T_i^- \approx n_i^+ - n_i^-. \tag{40}$$

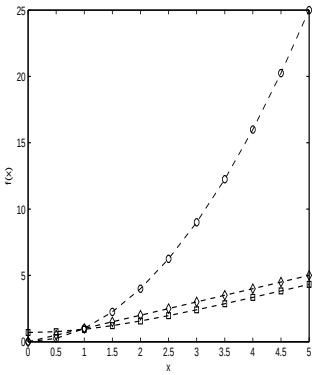
One considers the ratio, which is independent from the values of n_i's, but the other considers the difference, whose value scales with those of n_i's. Therefore, the estimate by NM-S.ML may be more biased than Ext-B.ML to fit comparison outcomes with large n_i.

Another issue is the small but perceivable dissimilarity of the ranking by Ext-B.RLS from those by Ext-B.ML and NM-S.ML, as revealed in the empirical GREs in Table 2 and the Kendall's tau in Table 4. Investigating them more carefully, we find that

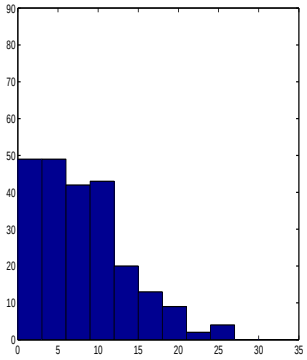
$$|d(\text{Ext-B.RLS}, \{\text{Ext-B.ML, NM-S.ML}\}) \geq 10| = 8, \tag{41}$$

where the distance is defined in (37). Interestingly, five of these eight partnerships played matches where weak teams beat strong teams by an extreme amount, such as Netherlands beating U.S.A.2 by 25:0, and Ext-B.RLS ranks them higher than Ext-B.ML and NM-S.ML do. This result suggests that Ext-B.RLS is vulnerable to even only few extreme outcomes so as to change the overall ranking. We verify this property by comparing the estimates by Ext-B.RLS and NM-S.ML. Suppose $n_i = n \; \forall i$ (which is the case here), and then according to (16), the ability estimate of individual s by Ext-B.RLS is

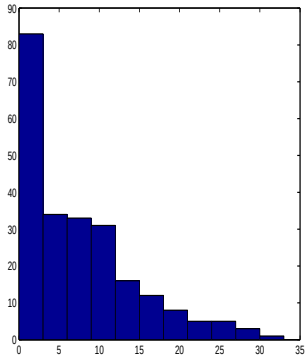
$$v_s = \sum_{i=1}^{m} A_{si}\big(\log n_i^+ - \log n_i^-\big) = \sum_{i=1}^{m} A_{si}\big(\log n_i^+ - \log(n - n_i^+)\big),$$



(a) Loss function curves. Circles: x^2. Diamonds: x. Squares: $\log(1+\cosh(x))$.



(b) Error histogram for NM-S.ML



(c) Error histogram for Ext-S.ML

Figure 5: Error function curves and histograms. The x-axis of histograms is $|T_i^+ - T_i^- - (n_i^+ - n_i^-)|$.

where $A = \left(G^T G + \mu I\right)^{-1} G^T$. To check the sensitivity of v_s with respect to the change of n_i^+, we calculate

$$\frac{\partial v_s}{\partial n_i^+} = A_{si}\left(\frac{1}{n_i^+} + \frac{1}{n - n_i^+}\right) = \frac{nA_{si}}{n_i^+(n - n_i^+)}.$$

Clearly, the estimate v_s is more sensitive to extreme values of n_i^+, that is, $n_i^+ \approx 0$ or $n_i^+ \approx n$. However, for NM-S.ML we have

$$v_s = \sum_{i=1}^{m} A_{si}(n_i^+ - n_i^-) = \sum_{i=1}^{m} A_{si}(2n_i^+ - n)$$

and

$$\frac{\partial v_s}{\partial n_i^+} = 2A_{si}.$$

That is, different values of n_i^+ have equal impact on the estimate by NM-S.ML.

In conclusion, when n_i remains a constant and the estimates by Ext-B.ML have $T_i^+ \approx T_i^- \ \forall i$, NM-S.ML and Ext-B.ML give similar estimates, which are less sensitive than that by Ext-B.RLS to extreme outcomes. When n_i's are unequal, the discussion in (40) indicates that NM-S.ML is more affected than Ext-B.ML.

5.2 Comparing the Two Scored-outcome Approaches

As shown in (38), the ranking by Ext-S.ML is rather diverse from those by Ext-B.RLS, Ext-B.ML, and NM-S.ML. We explore this issue by first re-writing the objective functions of NM-S.ML and Ext-S.ML respectively as

$$\min_{\mathbf{v}} \quad \sum_{i=1}^{m}\left(T_i^+ - T_i^- - (n_i^+ - n_i^-)\right)^2 + \mu\sum_{s=1}^{k} v_s^2$$

and

$$\min_{\mathbf{v}} \quad \sum_{i=1}^{m} \log\left(1+\cosh\left(T_i^+ - T_i^- - (n_i^+ - n_i^-)\right)\right) + \mu \sum_{s=1}^{k} (e^{v_s} + e^{-v_s}),$$

where cosh is the *hyperbolic cosine* function. Although these two formulations are derived to maximize the likelihood, they can be viewed as minimizing estimation errors

$$\left|T_i^+ - T_i^- - (n_i^+ - n_i^-)\right|$$

with two different loss functions. As μ is small, we ignore the effect of the regularization term. It is easy to show that as $\left|T_i^+ - T_i^- - (n_i^+ - n_i^-)\right| \to \infty$,

$$\frac{\log\left(1+\cosh\left(T_i^+ - T_i^- - (n_i^+ - n_i^-)\right)\right)}{\left|T_i^+ - T_i^- - (n_i^+ - n_i^-)\right|} \to 1.$$

To show the behaviors of the three functions: x^2, x and $\log(1+\cosh(x))$, we plot their curves in Figure 5(a). One can see that $\log(1+\cosh(x))$ increases almost linearly with x. In the machine learning community, it is well known that quadratic loss functions may lead to a very different estimation from linear ones. The reason is that quadratic loss functions penalize large errors more severely than linear ones do; estimations are thus dominated by even only few extreme observations, and as a side effect, may cause quite a few moderate errors. In contrast, estimations under linear loss functions may allow several large errors in order to make most errors small. Figures 5(b) and 5(c) are histograms of $\left|T_i^+ - T_i^- - (n_i^+ - n_i^-)\right|$ for NM-S.ML and Ext-S.ML, respectively; we see clearly the aforementioned two error patterns: Compared with NM-S.ML, Ext-S.ML has a lot more errors in the first bin and also some in the last two. In addition, we find that the empirical GRE of Ext-S.ML in Section 4.2 is highly related to its error pattern: Among the 24 correct rank predictions[7] produced by Ext-S.ML but not by NM-S.ML, twelve have $\left|T_i^+ - T_i^- - (n_i^+ - n_i^-)\right|$ smaller than 3 (the first bin of histograms); NM-S.ML has no $\left|T_i^+ - T_i^- - (n_i^+ - n_i^-)\right|$ larger than 27 (the last two bins of histograms) while Ext-S.ML has four, among which the partnerships satisfying (38) participate in three. Interestingly, the two types of loss functions seem to reflect two different ranking criteria: one focuses more on performances against extreme opponents, so wins over strong opponents and losses to weak opponents greatly influence the ranking; the other is less sensitive to extreme outcomes and treat comparisons more evenly. Consequently, deciding which loss function, and hence which approach to use may eventually be contingent on game-specific factors and subjective preferences.

6. Multi-class Classification

Multi-class classification using coding matrices (Dietterich and Bakiri, 1995; Allwein et al., 2001) is a general scheme to decompose a problem into several two-class problems. The widely-used methods "one-against-one" and "one-against-the rest" are special cases of this framework. The decomposition is usually specified by a coding matrix $G \in \{1,0,-1\}^{m\times k}$, where k is the number of classes and m is the number of two-class problems. Each row of G describes how the k classes are separated to two groups: those with 1 are in one group while those with -1 are in the other; those with 0 are not used in this two-class problem. The coding matrix in Table 5 illustrates four common types of codes: one-against-one, one-against-all, dense, and sparse; their definitions are

7. Correct rank predictions are the denominator of GRE minus its numerator.

One-against-one	$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 1 & 0 & 0 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$
One-against-the rest	
Dense	
Sparse	
$\vdots$	

Table 5: A coding matrix ($k = 8$). The four rows illustrate four types of codes.

given by Items 1 to 4 on Page 2210. At the training stage, m binary classifiers are trained for the m two-class problems. For an unlabeled instance, its label is predicted by combining results of the m binary classifiers.

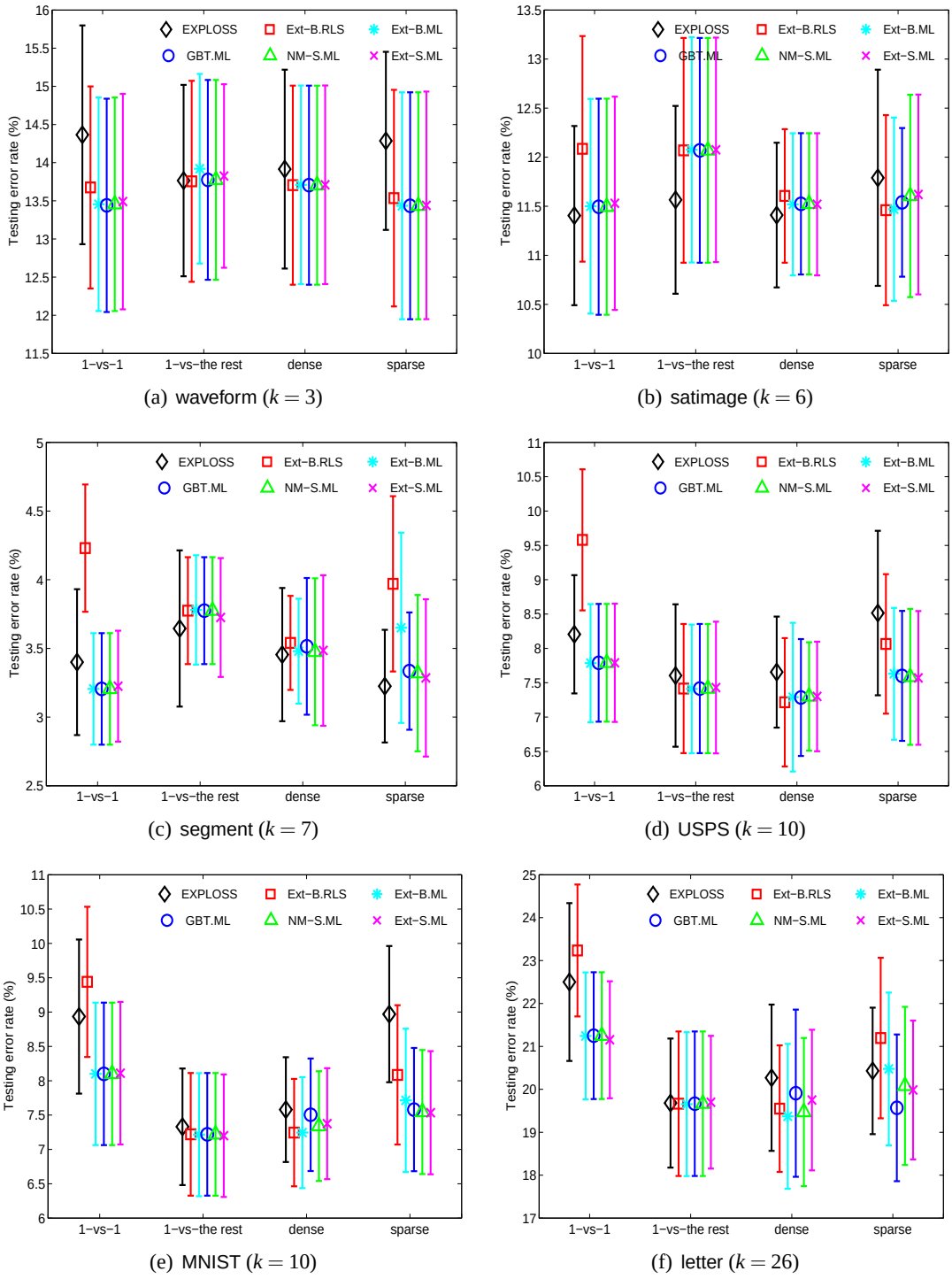
There are several schemes for deciding the final prediction. Dietterich and Bakiri (1995) proposed choosing the class whose column in G has the smallest distance to the m binary decisions on the instance. This method can correct errors made by some decision rules, and thus is called *error-correcting output codes* (ECOC). Allwein et al. (2001) proposed a more general framework, the *loss-based decoding*, which exploits not only binary decisions, but also *decision values* of binary classifiers. In particular, they adopted the *exponential loss-based decoding* (EXPLOSS): let $\hat{f}_i$ be the decision function of the ith binary classifier, and $\hat{f}_i(\mathbf{x}) > 0$ (< 0) specifies that an instance $\mathbf{x}$ to be in classes of I_i^+ (I_i^-). Then,

$$\text{predicted label } \equiv \arg\min_s \Big(\sum_{i=1}^{m} e^{-G_{is}\hat{f}_i} \Big).$$

If $G_{is} = 1$ and $\hat{f}_i(\mathbf{x})$ says $s \in I_i^+$, then $e^{-G_{is}\hat{f}_i}$ gives a small loss. By using decision values, the loss-based decoding incorporates the confidence of each binary prediction in making the final decision.

Table 5 is in the same format as our "comparison setting matrix" G defined in (12) and (13). Huang et al. (2006b) (GBT.ML) thus consider classes as individuals and two-class problems as group comparisons; the 1's and -1's in the ith row of G correspond to I_i^+ and I_i^-, respectively. The group competition results n_i^+ and n_i^- are assumed to be available from two-class classifiers. For an unlabeled instance, classes are ranked according to their estimated "abilities" and the highest one (with the largest ability) serves as the prediction. All of our newly proposed models can be applied in the same way, but there are two minor issues. Firstly, all of our proposed methods except Ext-B.RLS assume that group comparisons are independent. This property does not hold for multi-class classification since two-class classifiers involving the same classes share training data. Huang et al. (2006b) pointed out that GBT.ML can be interpreted as minimizing the Kullback-Leibler distance between the model and the observations. It is easy to see that their argument also applies to Ext-B.ML but not to NM-S.ML nor Ext-S.ML. Secondly, the n_i^+ and n_i^- given by two-class classifiers are real values, for which the binary-outcome approaches, according to their definition, may not be suitable. Despite of these minor issues, as we will show, our proposed methods perform quite well in practice.

We compare our methods with EXPLOSS and GBT.ML on six real data sets: waveform, satimage, segment, USPS, MNIST, and letter; numbers of classes range from 3 to 26. The settings of



(a) waveform ($k = 3$) (b) satimage ($k = 6$)

(c) segment ($k = 7$) (d) USPS ($k = 10$)

(e) MNIST ($k = 10$) (f) letter ($k = 26$)

Figure 6: Testing error rates on the 800-training-1000-testing data sets by six approaches under four codes: one-against-one (1-vs-1), one-against-the rest (1-vs-the rest), dense, and sparse. Vertical bars indicate standard deviations.

experiments are the same as those in Huang et al. (2006b). We use the 20 subsets of 800 training and 1,000 testing instances[8] and consider the same four types of coding matrices:

1. One-against-one: $|I_i^+| = |I_i^-| = 1,\ i = 1,\ldots,k(k-1)/2$.
2. One-against-all: $|I_i^+| = 1,\ |I_i^-| = k-1,\ i = 1,\ldots,k$.
3. Dense: $|I_i^+| = |I_i^-| = k/2,\ \forall i;\ m = [10\log_2 k]$.
4. Sparse: $E(|I_i^+|) = E(|I_i^-|) = k/4,\ \forall i;\ m = [15\log_2 k]$.

$[x]$ rounds a real number x to its nearest integer. We choose support vector machines (SVM) (Boser et al., 1992) with the RBF (Radial Basis Function) kernel $e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$ as the two-class classifier, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are two training instances. An improved version (Lin et al., 2007) of Platt (2000) generates n_i^+ and $n_i^- = 1 - n_i^+$ from SVM decision values. We implement our methods by modifying LIBSVM (Chang and Lin, 2001). For all of the 20 subsets, we select SVM parameters by cross validation before testing. Figures 6(a)-6(f) report the average testing error rates and standard deviations of the six methods: EXPLOSS, Ext-B.RLS, Ext-B.ML, GBT.ML, NM-S.ML and Ext-S.ML. Each figure summarizes the results on one data set by six groups of colored error bars, which represent the error rates of the six methods under the four types of codes. We can see that EXPLOSS (black diamond) and Ext-B.RLS (red square) perform worse than the others under the one-against-one and the sparse codes as k becomes large, while GBT.ML, Ext-B.ML, NM-S.ML and Ext-S.ML are almost equally good. Regarding the performances of the four types of codes, one-against-one and sparse are less effective for large values of k, an observation consistent with the results in (Huang et al., 2006b). Recall that in Section 4.2 Ext-S.ML behaves differently from the others, but here its predictions are similar to those of NM-S.ML and Ext-B.ML. The reason is that the n_i^+ and n_i^- produced by (Lin et al., 2007) are probabilities satisfying $n_i^+ + n_i^- = 1$, so values of $|T_i^+ - T_i^- - (n_i^+ - n_i^-)|$ are mostly small and the difference between quadratic and linear loss functions is negligible. Results here suggest that the proposed methods are useful for multi-class classification with coding matrices.

7. Conclusions

We propose new and useful methods to rank individuals from group comparisons. For comparisons with binary outcomes, earlier work solves non-convex problems, but here convex formulations with easy solution procedures are developed. For scored outcomes, our formulations are probably the first for this type of problems. Experiments show that the proposed approaches give reasonable partnership rankings from bridge records and perform effectively in multi-class classification. We give theoretical accounts for behaviors of proposed approaches, which demonstrate how different models reflect diverse ranking criteria. We also develop techniques to evaluate different rankings, which may be used in other ranking tasks.

Appendix A. Derivation of (10) from (8)

$$P(Y_i^+ - Y_i^- > 0) \equiv \int_{-\infty}^{\infty}\int_{y^-}^{\infty} de^{-e^{-(y^+-T_i^+)}} de^{-e^{-(y^--T_i^-)}}. \tag{42}$$

8. Available at http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob/data.

Let

$$x^+ \equiv e^{-(y^+ - T_i^+)} \text{ and } x^- \equiv e^{-(y^- - T_i^-)}.$$

Consequently,

$$de^{-e^{-(y^+ - T_i^+)}} = -e^{-x^+}dx^+ \text{ and } de^{-e^{-(y^- - T_i^-)}} = -e^{-x^-}dx^-.$$

Then,

$$\begin{aligned} (42) &= \int_0^\infty -e^{-x^-} \int_0^{x^- e^{T_i^+ - T_i^-}} -e^{-x^+}dx^+dx^- \\ &= \frac{e^{T_i^+}}{e^{T_i^+} + e^{T_i^-}}. \end{aligned}$$

Appendix B. Proof of Theorem 1

If $rank(G) < k$, G^TG is obviously not invertible; if $rank(G) = k$, the Singular Value Decomposition of G can be written as

$$G = U\Lambda V^T,$$

where $U \in R^{m\times k}$ and $V \in R^{k\times k}$ are orthonormal and $\Lambda \in R^{k\times k}$ is diagonal with

$$\Lambda_{ii} \neq 0,\ i = 1,\ldots,k.$$

Therefore,

$$G^TG = V\Lambda U^TU\Lambda V^T = V\Lambda^2V^T$$

is invertible.

Appendix C. Proof of Theorem 2

We first rewrite $l(\mathbf{v})$ as

$$l(\mathbf{v}) = -\sum_{i=1}^m (n_i^+T_i^+ + n_i^-T_i^-) + \sum_{i=1}^m n_i \log(e^{T_i^+} + e^{T_i^-}).$$

The first summation is obviously convex. For the second summation, by using Hölder's inequality we have

$$\begin{aligned} & \sum_{i=1}^m n_i \log\left(e^{\lambda T_i^+ + (1-\lambda)\tilde{T}_i^+} + e^{\lambda T_i^- + (1-\lambda)\tilde{T}_i^-}\right) \\ = & \sum_{i=1}^m n_i \log\left((e^{T_i^+})^\lambda (e^{\tilde{T}_i^+})^{1-\lambda} + (e^{T_i^-})^\lambda (e^{\tilde{T}_i^-})^{1-\lambda}\right) \\ \leq & \sum_{i=1}^m n_i \log\left(e^{T_i^+} + e^{T_i^-}\right)^\lambda \left(e^{\tilde{T}_i^+} + e^{\tilde{T}_i^-}\right)^{1-\lambda} \\ = & \sum_{i=1}^m n_i\lambda \log\left(e^{T_i^+} + e^{T_i^-}\right) + \\ & \sum_{i=1}^m n_i(1-\lambda) \log\left(e^{\tilde{T}_i^+} + e^{\tilde{T}_i^-}\right) \end{aligned} \quad (43)$$

for any $\mathbf{v}, \tilde{\mathbf{v}}$ and $\lambda \in (0,1)$, and the equality holds if and only if

$$T_i^+ - T_i^- = \tilde{T}_i^+ - \tilde{T}_i^- \ \forall i,$$

which can be re-written as

$$G(\mathbf{v} - \tilde{\mathbf{v}}) = \mathbf{0}. \tag{44}$$

If $rank(G) = k$, then (44) holds if and only if $\mathbf{v} = \tilde{\mathbf{v}}$, so $l(\mathbf{v})$ is strictly convex. If $l(\mathbf{v})$ is strictly convex, then the equality in (43) holds if and only if $\mathbf{v} = \tilde{\mathbf{v}}$, so

$$G(\mathbf{v} - \tilde{\mathbf{v}}) = \mathbf{0} \Leftrightarrow \mathbf{v} = \tilde{\mathbf{v}},$$

which implies $rank(G) = k$.

Appendix D. Proof of Theorem 3

It is easy to verify that the level sets of $l(\mathbf{v})$ are bounded. Since $l(\mathbf{v})$ is strictly convex, it then attains a unique global minimum. To prove the convergence of Algorithm 1, we first show that if $\partial l(\mathbf{v})/\partial v_s \neq 0$, then minimizing (22) leads to

$$l(\mathbf{v} + \delta) < l(\mathbf{v}). \tag{45}$$

From (23), if the optimal δ_s for (22) is zero, then

$$\frac{B_s + \sqrt{B_s^2 + 4\mu A_s e^{-v_s}}}{2A_s} = 1,$$

which implies

$$4A_s(\mu e^{-v_s} - A_s + B_s) = -4A_s \frac{\partial l(\mathbf{v})}{\partial v_s} = 0. \tag{46}$$

Since $A_s \neq 0$ throughout iterations, (46) implies $\partial l(\mathbf{v})/\partial v_s = 0$. Thus if $\partial l(\mathbf{v})/\partial v_s \neq 0$, the optimal $\delta_s \neq 0$. With (22) $= 0$ if $\delta_s = 0$, (45) follows.

Next we show that the sequence $\{\mathbf{v}^t\}$ generated by Algorithm 1 is bounded. If not, there must exist j such that $|v_j^t| \to \infty$. Then

$$\begin{aligned} l(\mathbf{v}^t) &\geq \mu \sum_{s=1}^{k} (e^{v_s^t} + e^{-v_s^t}) \\ &= \mu \sum_{s=1}^{k} (e^{|v_s^t|} + e^{-|v_s^t|}) \\ &\geq \mu e^{|v_j^t|} + e^{-|v_j^t|} \\ &\to \infty, \end{aligned}$$

which contradicts the fact that

$$l(\mathbf{v}^0) > l(\mathbf{v}^t) \ \forall t.$$

Since $\{\mathbf{v}^t\}$ is bounded, it has limit points. For any limit point $\mathbf{v}^*$, there is an infinite set $\bar{N}$ such that

$$\lim_{t \in \bar{N}, t \to \infty} \mathbf{v}^t = \mathbf{v}^*.$$

Since $\mathbf{v}$ is finite dimensional, there is one component s updated in an infinite set $N \subset \bar{N}$:

$$(t \bmod k) + 1 = s \text{ for } t \in N.$$

Because $l(\mathbf{v})$ is convex, to prove that $\mathbf{v}^*$ is a global minimum, it suffices to show that

$$\frac{\partial l(\mathbf{v}^*)}{\partial v_s} = 0 \text{ for } s = 1, \ldots, k. \tag{47}$$

Suppose the contrary is true, then among $s, s+1, \ldots, k, 1, \ldots, s-1$, there is $\bar{s}$ such that

$$\frac{\partial l(\mathbf{v}^*)}{\partial v_s} = \cdots = \frac{\partial l(\mathbf{v}^*)}{\partial v_{\bar{s}-1}} = 0, \ \frac{\partial l(\mathbf{v}^*)}{\partial v_{\bar{s}}} \neq 0. \tag{48}$$

From (45), updating $v_{\bar{s}}^*$ by (23) yields $\mathbf{v}^{*+1} \neq \mathbf{v}^*$ and

$$l(\mathbf{v}^{*+1}) < l(\mathbf{v}^*).$$

We have that $\partial l(\mathbf{v}^*)/\partial v_s = 0$ implies

$$\frac{B_s + \sqrt{B_s^2 + 4\mu A_s^* e^{-v_s^*}}}{2A_s^*} = 1,$$

where A_s^* is defined according to (24) and B_s is a constant independent of $\mathbf{v}$. Therefore,

$$\begin{aligned} \lim_{\substack{t \in N, \\ t \to \infty}} v_s^{t+1} &= \lim_{\substack{t \in N, \\ t \to \infty}} \left(v_s^t + \log \frac{B_s + \sqrt{B_s^2 + 4\mu A_s^t e^{-v_s^t}}}{2A_s^t} \right) \\ &= v_s^* + \log \frac{B_s + \sqrt{B_s^2 + 4\mu A_s^* e^{-v_s^*}}}{2A_s^*} \\ &= v_s^*, \end{aligned} \tag{49}$$

and

$$\lim_{t \in N, t \to \infty} \mathbf{v}^{t+1} = \lim_{t \in N, t \to \infty} \mathbf{v}^t = \mathbf{v}^*. \tag{50}$$

Let $\bar{t}$ be the iteration corresponding to $\bar{s}$. Using (48), a similar derivation to (49) and (50) shows that

$$\lim_{\substack{t \in N, \\ t \to \infty}} \mathbf{v}^{t+1} = \cdots = \lim_{\substack{t \in N, \\ t \to \infty}} \mathbf{v}^{\bar{t}} = \mathbf{v}^* \text{ and } \lim_{\substack{t \in N, \\ t \to \infty}} \mathbf{v}^{\bar{t}+1} = \mathbf{v}^{*+1};$$

consequently,

$$\lim_{t \in N, t \to \infty} l(\mathbf{v}^{\bar{t}+1}) = l(\mathbf{v}^{*+1}) < l(\mathbf{v}^*),$$

which contradicts the fact that

$$l(\mathbf{v}^*) \leq \cdots \leq l(\mathbf{v}^{t+1}) \leq l(\mathbf{v}^t).$$

Thus (47) holds for all limit points. Since $l(\mathbf{v})$ is strictly convex, every limit point is the unique global minimum. Moreover, the sequence $\{\mathbf{v}^t\}$ is bounded, so it globally converges to the global minimum.

Appendix E. Proof of Claim 1

From (29) it is clear that the ranking by NM-S.ML is invariant to the scale of n_i; we thus assume

$$n_i^+ + n_i^- = 2, \forall i.$$

Then (26) can be rewritten as

$$\min_{\mathbf{v}} \sum_{i=1}^{m} \left((T_i^+ - T_i^-)^2 - (4n_i^+ - 4)(T_i^+ - T_i^-) \right).$$

For Ext-B.ML, as μ is small and can be ignored, we consider the objective function in (17), which can be re-written as

$$\sum_{i=1}^{m} -n_i^+ (T_i^+ - T_i^-) + n_i \log(e^{T_i^+ - T_i^-} + 1) \tag{51}$$

$$= \sum_{i=1}^{m} -n_i^+ (T_i^+ - T_i^-) + 2\Big(\log 2 + \frac{1}{2}(T_i^+ - T_i^-) + \frac{1}{8}(T_i^+ - T_i^-)^2 + O\big((T_i^+ - T_i^-)^3\big) \Big) \tag{52}$$

$$\approx \frac{1}{8} \sum_{i=1}^{m} \left((T_i^+ - T_i^-)^2 - (4n_i^+ - 4)(T_i^+ - T_i^-) \right).$$

From (51) to (52) we use the Taylor expansion of the function $\log(e^x + 1)$ at $x = 0$ and the assumption that $T_i^+ \approx T_i^- \ \forall i$. Therefore, the rankings by NM-S.ML and Ext-B.ML are similar.

Appendix F. Top 10 Partnerships by Ext-B.ML

Team	Players	
U.S.A.2	Eric Greco	Geoff Hampson
Sweden	Peter Bertheau	Fredrik Nystrom
Japan	Yoshiyuki Nakamura	Yasuhiro Shimizu
Chinese Taipei	Chih-Kuo Shen	Jui-Yiu Shih
New Zealand	Tom Jacob	Malcolm Mayer
China	Zhong Fu	Jie Zhao
Italy	Norberto Bocchi	Giorgio Duboin
Brazil	Gabriel Chagas	Miguel Villas-boas
India	Subhash Gupta	Rajeshwar Tewari
Portugal	Jorge Castanheira	Sofia Pessoa

References

Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001. ISSN 1533-7928.

Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

Ralph A. Bradley and Milton E. Terry. The rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324–345, 1952.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

John N. Darroch and Douglas Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

Herbert A. David. *The Method of Paired Comparisons*. Oxford University Press, second edition, 1988.

Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 2nd edition, 1986.

Mark E. Glickman. *Paired Comparison Models with Time-varying Parameters*. PhD thesis, Department of Statistics, Harvard University, 1993.

Joshua Goodman. Sequential conditional generalized iterative scaling. In *ACL*, pages 9–16, 2002.

Trevor Hastie and Robert Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

Ralf Herbrich and Thore Graepel. TrueSkill$^{\text{TM}}$: A Bayesian skill rating system. In *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

Tzu-Kuo Huang, Chih-Jen Lin, and Ruby C. Weng. Ranking individuals by group comparisons. In *Proceedings of the Twenty Third International Conference on Machine Learning (ICML)*, 2006a.

Tzu-Kuo Huang, Ruby C. Weng, and Chih-Jen Lin. Generalized Bradley-Terry models and multi-class probability estimates. *Journal of Machine Learning Research*, 7:85–115, 2006b. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/generalBT.pdf`.

David R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32:386–408, 2004.

Edwin T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620–630, 1957a.

Edwin T. Jaynes. Information theory and statistical mechanics ii. *Physical Review*, 108(2):171–190, 1957b.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68:267–276, 2007. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/plattprob.pdf`.

Joshua E. Menke and Tony R. Martinez. A Bradley-Terry artificial neural network model for individual ratings in group competitions. *Neural Computing and Applications*, 2007. To appear.

Thomas Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, MIT, 2001.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

John Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press.

Bianca Zadrozny. Reducing multiclass to binary by coupling probability estimates. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1041–1048. MIT Press, Cambridge, MA, 2002.

Journal of Machine Learning Research 9 (2008) 2217-2250 Submitted 6/08; Revised 8/08; Published 10/08

Forecasting Web Page Views: Methods and Observations

Jia Li JIALI@STAT.PSU.EDU
Visiting Scientist∗
Google Labs, 4720 Forbes Avenue
Pittsburgh, PA 15213

Andrew W. Moore AWM@GOOGLE.COM
Engineering Director
Google Labs, 4720 Forbes Avenue
Pittsburgh, PA 15213.

Editor: Lyle Ungar

Abstract

Web sites must forecast Web page views in order to plan computer resource allocation and estimate upcoming revenue and advertising growth. In this paper, we focus on extracting trends and seasonal patterns from page view series, two dominant factors in the variation of such series. We investigate the Holt-Winters procedure and a state space model for making relatively short-term prediction. It is found that Web page views exhibit strong impulsive changes occasionally. The impulses cause large prediction errors long after their occurrences. A method is developed to identify impulses and to alleviate their damage on prediction. We also develop a long-range trend and season extraction method, namely the *Elastic Smooth Season Fitting (ESSF)* algorithm, to compute scalable and smooth yearly seasons. ESSF derives the yearly season by minimizing the residual sum of squares under smoothness regularization, a quadratic optimization problem. It is shown that for long-term prediction, ESSF improves accuracy significantly over other methods that ignore the yearly seasonality.

Keywords: web page views, forecast, Holt-Winters, Kalman filtering, elastic smooth season fitting

1. Introduction

This is a machine learning application paper about a prediction task that is rapidly growing in importance: predicting the number of visitors to a Web site or page over the coming weeks or months. There are three reasons for this growth in importance. First, hardware and network bandwidth need to be provisioned if a site is growing. Second, any revenue-generating site needs to predict its revenue. Third, sites that sell advertising space need to estimate how many page views will be available before they can commit to a contract from an advertising agency.

1.1 Background on Time Series Modeling

Time series are commonly decomposed into "trend", "season", and "noise":

$$X_t = L_t + I_t + N_t \ , \tag{1}$$

∗. Also, Associate Professor, Department of Statistics, The Pennsylvania State University.

where L_t is trend, I_t is season, and N_t is noise. For some prediction methods, L_t is more than a global growth pattern, in which case it will be referred to as "level" to distinguish from the global pattern often called trend. These components of a time series need to be treated quite differently. The noise N_t is often modeled by stationary ARMA (autoregressive moving average) process (Brockwell and Davis, 2002; Wei, 2006). Before modeling the noise, the series needs to be "detrended" and "deseasoned". There are multiple approaches to trend and season removal (Brockwell and Davis, 2002). In the well-known Box-Jenkins ARIMA (autoregressive integrated moving average) model (Box and Jenkins, 1970), the difference between adjacent lags (i.e., time units) is taken as noise. The differencing can be applied several times. The emphasis of ARIMA is still to predict noise. The trend is handled in a rather rigid manner (i.e., by differencing). In some cases, however, trend and season may be the dominant factors in prediction and require methods devoted to their extraction. A more sophisticated approach to compute trend is by smoothing, for instance, global polynomial fitting, local polynomial fitting, kernel smoothing, and exponential smoothing. Exponential smoothing is generalized by the Holt-Winters (HW) procedure to include seasonality. Chatfield (2004) provides practical accounts on when ARIMA model or methods aimed at capturing trend and seasonality should be used.

Another type of model that offers the flexibility of handling trend, season, and noise together is the state space model (SSM) (Durbin and Koopman, 2001). The ARIMA model can be cast into an SSM, but SSM includes much broader non-stationary processes. SSM and its computational method—the Kalman filter were developed in control theory and signal processing (Kalman, 1960; Sage and Melsa, 1971; Anderson and Moore, 1979). For Web page view series, experiments suggest that trend and seasonality are more important than the noise part for prediction. We thus investigate the HW procedure and an SSM emphasizing trend and seasonality. Despite its computational simplicity, HW has been successful in some scenarios (Chatfield, 2004). The main advantages of SSM over HW are (a) some parameters in the model are estimated based on the series, and hence the prediction formula is adapted to the series; (b) if one wants to modify the model, the general framework of SSM and the related computational methods apply the same way, while HW is a relatively specific static solution.

1.2 Web Page View Prediction

Web page view series exhibit seasonality at multiple time scales. For daily page view series, there is usually a weekly season and sometimes a long-range yearly season. Both HW and SSM can effectively extract the weekly season, but not the yearly season for several reasons elaborated in Section 4. For this task, we develop the Elastic Smooth Season Fitting (ESSF) method. It is observed that instead of being a periodic sequence, the yearly seasonality often emerges as a yearly pattern that may scale differently across the years. ESSF takes into consideration the scaling phenomenon and only requires two years of data to compute the yearly season. Experiments show that the prediction accuracy can be improved remarkably based on the yearly season computed by ESSF, especially for forecasting distant future.

To our best knowledge, existing work on forecasting Internet access data is mostly for network traffic load. For short-term traffic, it is reasonable to assume that the random process is stationary, and thus prediction relies on extracting the serial statistical dependence in the seemingly noisy series. Stationary ARMA models are well suited for such series and have been exploited (Basu et al., 1996; You and Chandra, 1999). A systematic study of the predictability of network traffic based

on stationary traffic models has been conducted by Sang and Li (2001). For long-term prediction of large-scale traffic, because trends often dominate, prediction centers around extracting trends. Depending on the characteristics of trends, different methods may be used. In some cases, trends are well captured by growth rate and the main concern is to accurately estimate the growth rate, for instance, that of the overall Internet traffic (Odlyzko, 2003). Self-similarity is found to exist at multiple time scales of network traffic, and is exploited for prediction (Grossglauser and Bolot, 1999). Multiscale wavelet decomposition has been used to predict one-minute-ahead Web traffic (Aussem and Murtagh, 2001), as well as Internet backbone traffic months ahead (Papagiannaki et al., 2005). Neural networks have also been applied to predict short-term Internet traffic (Khotanzad and Sadek, 2003). An extensive collection of work on modeling self-similar network traffic has been edited by Park and Willinger (2000).

We believe Web page view series, although closely related to network traffic data, have particular characteristics worthy of a focused study. The contribution of the paper is summarized as follows.

1. We investigate short-term prediction by HW and SSM. The advantages and disadvantages of the two approaches in various scenarios are analyzed. It is also found that seasonality exists at multiple time scales and is important for forecasting Web page view series.

2. Methods are developed to detect sudden massive impulses in the Web traffic and to remedy their detrimental impact on prediction.

3. For long-term prediction several months ahead, we develop the ESSF algorithm to extract global trends and scalable yearly seasonal effects after separating the weekly season using HW.

1.3 Application Scope

The prediction methods in this paper focus on extracting trend and season at several scales, and are not suitable for modeling stationary stochastic processes. The ARMA model, for which mature off-the-shelf software is available, is mostly used for such processes. The trend extracted by HM or SSM is the noise-removed non-season portion of a time series. If a series can be compactly described by a growth rate, it is likely better to directly estimate the growth rate. However, HW and SSM are more flexible in the sense of not assuming specific functional form for the trend on the observed series. HW and SSM are limited for making long-term prediction. By HW, the predicted level term of the page view at a future time is assumed to be the current level added by a linear function of the time interval, or simply the current level if linear growth is removed, as in some reduced form of HW. If a specifically parameterized function can be reliably assumed, it is better to estimate parameters in the function and apply extrapolation accordingly. However, in the applications we investigated, there is little base for choosing any particular function. The yearly season extraction by ESSF is found to improve long-term prediction. The basic assumption of ESSF is that the time series exhibits a yearly pattern, possibly scaled differently across the years. It is not intended to capture event driven pattern. For instance, the search volume for `Batman` surges around the release of every new Batman movie, but shows no clear yearly pattern.

In particular, we have studied two types of Web page view series: (a) small to moderate scale Web sites; (b) dynamic Web pages generated by Google for given search queries. Due to the fast changing pace of the Internet, page view series available for small to moderate scale Web sites are usually short (e.g., shorter than two years). Therefore, the series are insufficient for exploiting

yearly seasonality in prediction. The most dramatic changes in those series are often the news-driven surges. Without side information, such surges cannot be predicted from the page view series alone. It is difficult for us to acquire page view data from Web sites with long history and very high access volume because of privacy constraints. We expect the page views of large-scale Web sites to be less impulsive in a relative sense because of their high base access. Moreover, large Web sites are more likely to have existed long enough to form long-term, for example, yearly, access patterns. Such characteristics are also possessed by the world-wide volume data of search queries, which we use in our experiments.

The rest of the paper is organized as follows. In Section 2, the Holt-Winters procedure is introduced. The effect of impulses on the prediction by HW is analyzed, based on which methods of detection and correction are developed. In Section 3, we present the state space model and discuss the computational issues encountered. Both HW and SSM aim at short-term prediction. The ESSF algorithm for long-term prediction is described in Section 4. Experimental results are provided in Section 5. We discuss predicting the noise part of the series by AR (autoregressive) models and finally conclude in Section 6.

2. The Holt-Winters Procedure

Let the time series be $\{x_1, x_2, ..., x_n\}$. The Holt-Winters (HW) procedure (Chatfield, 2004) decomposes the series into level L_t, season I_t, and noise. The variation of the level after one lag is assumed to be captured by a local linear growth term T_t. Let the period of the season be d. The HW procedure updates L_t, I_t, and T_t simultaneously by a recursion:

$$\begin{aligned} L_t &= \zeta(x_t - I_{t-d}) + (1-\zeta)(L_{t-1} + T_{t-1}), && (2)\\ T_t &= \kappa(L_t - L_{t-1}) + (1-\kappa)T_{t-1}, && (3)\\ I_t &= \delta(x_t - L_t) + (1-\delta)I_{t-d} && (4) \end{aligned}$$

where the pre-selected parameters $0 \le \zeta \le 1$, $0 \le \kappa \le 1$, and $0 \le \delta \le 1$ control the smoothness of updating. This is a stochastic approximation method in which the current level is an exponentially weighted running average of recent season-adjusted observations. To better see this, let us assume the season and linear growth terms are absent. Then Eq. (2) reduces to

$$\begin{aligned} L_t &= \zeta x_t + (1-\zeta)L_{t-1} && (5)\\ &= \zeta x_t + (1-\zeta)\zeta x_{t-1} + (1-\zeta)^2 L_{t-2}\\ &\;\;\vdots\\ &= \zeta x_t + (1-\zeta)\zeta x_{t-1} + (1-\zeta)^2\zeta x_{t-2} + \cdots + (1-\zeta)^{t-1}\zeta x_1 + (1-\zeta)^t L_0. \end{aligned}$$

Suppose L_0 is initialized to zero, the above equation is an on the fly exponential smoothing of the time series, that is, a weighted average with the weights attenuating exponentially into the past. We can also view L_t in Eq. (5) as a convex combination of the level indicated by the current observation x_t and the level suggested by the past estimation L_{t-1}. When the season is added, x_t subtracted by the estimated season at t becomes the part of L_t indicated by current information. At this point of recursion, the most up-to-date estimation for the season at t is I_{t-d} under period d. When the linear growth is added, the past level L_{t-1} is expected to become $L_{t-1} + T_{t-1}$ at t. Following the same scheme of convex combination, Eq. (5) evolves into (2). Similar rationale applies to the update

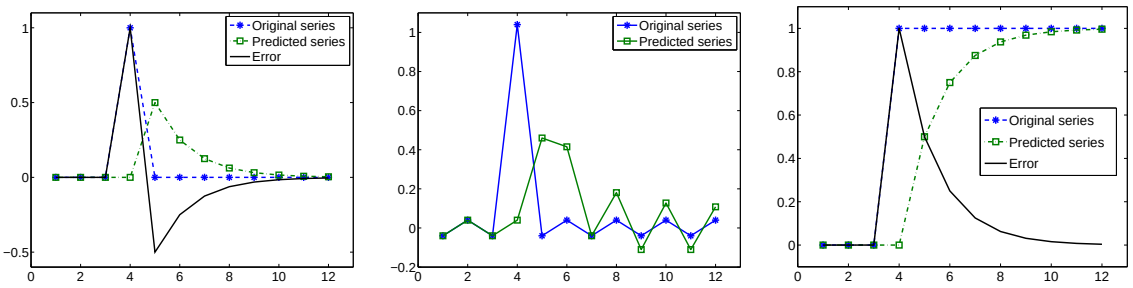


Figure 1: Holt-Winters prediction for time series with abrupt changes. (a) Impulse effect on a leveled signal: slow decaying tail; (b) Impulse effect on a periodic signal: ripple effect; (c) Response to a step signal.

of T_t and I_t in Eqs. (3) and (4). Based on past information, T_t and I_t are expected to be T_{t-1} and I_{t-d} under the implicit assumption of constant linear growth and fixed season. On the other hand, the current x_t and the newly computed L_t suggest T_t to be $L_t - L_{t-1}$, and I_t to be $x_t - L_t$. Applying convex combination leveraging past and current information, we obtain Eqs. (3) and (4).

To start the recursion in the HW procedure at time t, initial values are needed for L_{t-1}, T_{t-1}, and $I_{t-\tau}$, $\tau = 1,2,...,d$. We use the first period of data $\{x_1, x_2, ..., x_d\}$ for initialization, and start the recursion at $t = d+1$. Specifically, linear regression is conducted for $\{x_1, x_2, ..., x_d\}$ versus the time grid $\{1,2,...,d\}$. That is, x_τ and τ, $\tau = 1,...,d$, are treated as dependent variable and independent variable respectively. Suppose the regression function obtained is $b_1\tau + b_2$. We initialize by setting $L_\tau = b_1\tau + b_2$, $T_\tau = 0$, and $I_\tau = x_\tau - L_\tau$, $\tau = 1,2,...,d$.

The forecasting of h time units forward at t, that is, the prediction of x_{t+h} based on $\{x_1, x_2, ..., x_t\}$, is

$$\hat{x}_{t+h} = L_t + hT_t + I_{t-d+h \bmod d} ,$$

where *mod* is the modulo operation. The linear function of h, $L_t + hT_t$, with slope given by the most updated linear growth T_t, can be regarded as an estimation for L_{t+h}; while $I_{t-d+h \bmod d}$, the most updated season at the same cyclic position as $t+h$, which is already available at t, is the estimation for I_{t+h}.

Experiments using the HW procedure show that the local linear growth term, T_t, helps little in prediction. In fact, for relatively distant future, the linear growth term degrades performance. This is because for the Web page view series, we rarely see any linear trends visible over a time scale from which the gradient can be estimated by HW. We can remove the term T_t in HW conveniently by initializing it with zero and setting the corresponding smoothing parameter $\kappa = 0$.

Web page view series sometimes exhibit impulsive surges or dips. Such impulsive changes last a short period of time and often bring the level of page views to a magnitude one or several orders higher than the normal range. For instance, in Figure 5(a), the amount of page views for an example Web site jumps tremendously at the 404th day and returns to normal one day later. Impulses are triggered by external forces which are unpredictable based on the time series alone. One such common external force is a news launch related to the Web site. Because it is extremely difficult if possible at all to predict the occurrence of an impulse, we focus on preventing its after effect.

The influence of an impulse on the prediction by the HW procedure is elaborated in Figure 1. In Figure 1(a), a flat leveled series with an impulse is processed by HW. The predicted series attempts to catch up with the impulse after one lag. Although the impulse is over after one lag, the predicted series attenuates slowly, causing large errors several lags later. The stronger the impulse is, the slower the predicted series returns close to the original one. The prediction error consumes a positive value and then a negative one, both of large magnitudes. Apparently, a negative impulse will result in a reversed error pattern. Figure 1(b) shows the response of HW to an impulse added to a periodic series. The prediction error still yields the pattern of switching signs and large magnitudes. To reduce the influence of an impulse, it is important that we differentiate an impulse from a sudden step-wise change in the series. When a significant step appears, we want the predicted series to catch up with the change as fast as possible rather than hindering the strong response. Figure 1(c) shows the prediction by HW for a series with a sudden positive step change. The prediction error takes a large positive value and reduces gradually to zero without crossing into the negative side.

Based on the above observations, we detect an impulse by examining the co-existence of errors with large magnitudes and opposite signs within a short window of time. In our experiments, the window size is $s_1 = 10$. The extremity of the prediction error is measured relatively with respect to the standard deviation of prediction errors in the most recent past of a pre-selected length. In the current experiment, this length is $s_2 = 50$. The time units of s_1 and s_2 are the same as that of the time series in consideration. Currently, we manually set the values of $s_{1,2}$. The rationale for choosing these values is that s_1 implies the maximum length of an impulse; and s_2 balances accurate estimation of the noise variance and swift adaptation to the change of the variance over time. We avoid setting s_1 too high to ensure that a detected impulse is a short-lived, strong, and abrupt change. If a time series undergoes a real sudden rising or falling trend, the prediction algorithm will capture the trend but with a certain amount of delay, as shown by the response of HW to a step signal in Figure 1(c). In a special scenario when an impulse locates right at the boundary of a large rising trend, the measure taken to treat the impulse will further slow down the response to, but not prevent the eventual catch-up of the rise.

At time t, let the prediction for x_t based on the past series up to $t-1$ be $\hat{x}_t$, and the prediction error be $e_t = x_t - \hat{x}_t$. We check whether an impulse has started at t', $t - s_1 + 1 \le t' \le t - 1$, and ended at t by the following steps.

1. Compute the standard deviation with removed outliers, σ_{t-1}, for the prediction errors $\{e_{t-s_2}, e_{t-s_2+1}, ..., e_{t-1}\}$, which are known by time t. The motivation for removing the outliers is that at any time an impulse exists, the prediction error will be unusually large, and hence bias the estimated average amount of variation. In our experiments, 10% of the errors are removed as outliers.

2. Compute the relative magnitude of e_t by $\theta_t = \frac{|e_t|}{\sigma_{t-1}}$.

3. Examine $\theta_{t'}$ in the window $t' \in [t - s_1 + 1, t]$. If there is a t', $t - s_1 + 1 \le t' \le t - 1$, such that $\theta_{t'} > \Delta_1$ and $\theta_t > \Delta_2$ and $sign(e_{t'}) \neq sign(e_t)$, the segment $[t', t]$ is marked as an impulse. If $e_{t'}$ is positive while e_t negative, the impulse is a surge; the reverse is a dip. The two thresholds Δ_1 and Δ_2 determine the sensitivity to impulses and are chosen around 2.5.

If impulse is not detected, the HW recursion is applied at the next time unit $t+1$. Otherwise, L_t, T_t, and $I_{t'}$ for $t' \in [t - s_1 + 1, t]$, are revised as follows to reduce the effect of the impulse on the future L_τ, T_τ, and I_τ, $\tau > t$. Once the revision is completed, the HW recursion resumes at $t+1$.

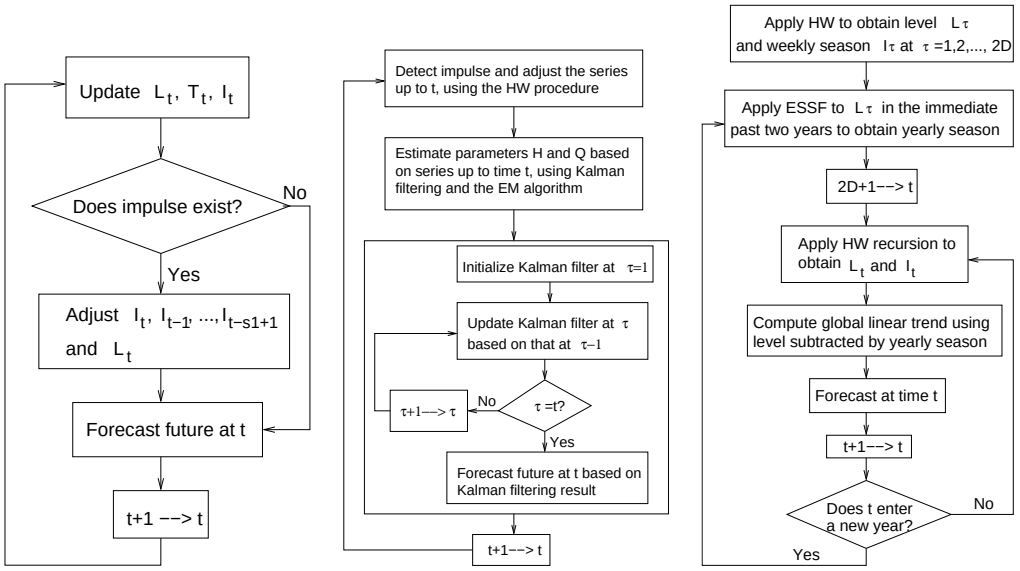


Figure 2: The schematic diagrams for the forecasting algorithms: (a) Holt-Winters with impulse detection; (b) GLS; (c) ESSF.

1. For $t' = t - s_1 + 1, \ldots, t$, set $I_{t'} = I_{t'-d}$ sequentially. This is equivalent to discarding the season computed during the impulse segment and using the most recent season right before the impulse.

2. Let $L_t = \frac{1}{2} L_{t-s_1} + \frac{1}{2}(x_t - I_t)$, where L_{t-s_1} is the level before the impulse and I_t is the already revised season at t.

3. Let $T_t = 0$.

In this paper, we constrain our interest to reducing the adverse effect of an impulse on later prediction after it has occurred and been detected. Predicting the arrival of impulses in advance using side information, for instance, scheduled events impacting Web visits, is expected to be beneficial, but is beyond our study here. A schematic diagram of the HW procedure is illustrated in Figure 2(a).

Holt-Winters and our impulse-resistant modification have the merit of being very cheap to update and predict, requiring only a handful of additions and multiples. This may be useful in some extremely high throughput situations, such as network routers. But in more conventional settings, it leads to the question: can we do better with more extensive model estimation at each time step?

3. State Space Model

A state space model (SSM) assumes that there is an underlying state process for the series $\{x_1, \ldots, x_n\}$. The states are characterized by a Markov process, and x_t is a linear combination of the states added

with Gaussian noise. In general, an SSM can be represented in the following matrix form:

$$\begin{aligned} x_t &= \mathbf{Z}_t\alpha_t + \varepsilon_t\,, \qquad \varepsilon_t \sim N(0,\mathbf{H}_t), \\ \alpha_{t+1} &= \mathbf{T}_t\alpha_t + \mathbf{R}_t\eta_t\,, \quad \eta_t \sim N(0,\mathbf{Q}_t)\,, \quad t = 1,...,n, \\ &\qquad\qquad \alpha_1 \sim N(a_1,\mathbf{P}_1) \end{aligned} \tag{6}$$

where $\{\alpha_1,\alpha_2,...,\alpha_n\}$ is the state process. Each state is an m-dimensional column vector. Although in our work, the observed series x_t is univariate, SSM treats generally p-dimensional series. The noise terms ε_t and η_t follow Gaussian distributions with zero mean and covariance matrices $\mathbf{H}_t$ and $\mathbf{Q}_t$ respectively. For clarity, we list the dimension of the matrices and vectors in (6) below.

observation	x_t	$p\times 1$	$\mathbf{Z}_t$	$p\times m$
state	α_t	$m\times 1$	$\mathbf{T}_t$	$m\times m$
noise	ε_t	$p\times 1$	$\mathbf{H}_t$	$p\times p$
noise	η_t	$r\times 1$	$\mathbf{R}_t$	$m\times r$
			$\mathbf{Q}_t$	$r\times r$
initial state mean	a_1	$m\times 1$	$\mathbf{P}_1$	$m\times m$

We restrict our interest to time invariant SSM where the subscript t can be dropped for $\mathbf{Z}$, $\mathbf{T}$, $\mathbf{R}$, $\mathbf{H}$, and $\mathbf{Q}$. Matrices $\mathbf{Z}$, $\mathbf{T}$ and $\mathbf{R}$ characterize the intrinsic relationship between the state and the observed series, as well as the transition between states. They are determined once we decide upon a model. The covariance matrices $\mathbf{H}$ and $\mathbf{Q}$ are estimated based on the time series using the Maximum Likelihood (ML) criterion.

Next, we describe the *Level with Season (LS)* model, which decomposes x_t in the same way as the HW procedure in Eq. (2)$\sim$(4), with the linear growth term removed. We discard the growth term because, as mentioned previously, this term does not contribute in the HW procedure under our experiments. However, if necessary, it would be easy to modify the SSM to include this term. We then describe the *Generalized Level with Season (GLS)* model that can explicitly control the smoothness of the level.

3.1 The Level with Season Model

Denote the level at t by μ_t and the season with period d by i_t. The LS model assumes

$$\begin{aligned} x_t &= \mu_t + i_t + \varepsilon_t, \\ i_t &= -\sum_{j=1}^{d-1} i_{t-j} + \eta_{1,t}, \\ \mu_t &= \mu_{t-1} + \eta_{2,t} \end{aligned} \tag{7}$$

where ε_t and $\eta_{j,t}$, $j = 1,2$, are the Gaussian noises.

Comparing with the HW recursion equations (2)$\sim$(4), Eq. (7) is merely a model specifying the statistical dependence of x_t on μ_t and i_t, both of which are unobservable random processes. The Kalman filter for this model, playing a similar role as Eqs. (2)-(4) for HW, will be computed recursively to estimate μ_t, i_t, and to predict future. Details on the Kalman filter are provided in Appendix A. In its simplest form, with both the linear growth and season term removed, HW reduces to exponential smoothing with recursion $L_t = \zeta x_t + (1-\zeta)L_{t-1}$. It can be shown that if we let $L_t = E(\mu_t \mid x_1,...,x_{t-1})$, the recursion for L_t in HW is the same as that derived from the Kalman

filter for the LS model without season. The smoothing parameter ζ is determined by the parameters of the noise distributions in LS. When season is added, there is no complete match between the recursion of HW and that of the Kalman filter. In the LS model, it is assumed that $\sum_{\tau=1}^{d} i_{t+\tau} = 0$ up to white noise, but HW does not enforce the zero sum of one period of the season terms. The decomposition of x_t into level μ_t and season i_t by LS is however similar to that assumed by HW.

We can cast the LS model into a time invariant SSM following the notation of (6). The matrix expansion according to (6) leads to the same set of equations in (7):

$$\alpha_t = \begin{pmatrix} i_t \\ i_{t-1} \\ \vdots \\ i_{t-d+2} \\ \mu_t \end{pmatrix}, \quad \eta_t = \begin{pmatrix} \eta_{1,t} \\ \eta_{2,t} \end{pmatrix},$$

$$\underset{d\times 1}{\mathbf{Z} = \quad (1,0,0,\cdots,0,1)}, \quad \underset{d\times 3}{\mathbf{R} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & 1 \end{pmatrix}}, \quad \underset{d\times d}{\mathbf{T} = \begin{pmatrix} -1 & -1 & -1 & \cdots & -1 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix}}.$$

3.2 Generalized Level with Season Model

We generalize the above LS model by imposing different extent of smoothness on the level term μ_t. Specifically, let

$$\begin{aligned} x_t &= \mu_t + i_t + \varepsilon_t, \\ i_t &= -\sum_{j=1}^{s-1} i_{t-j} + \eta_{1,t}, \\ \mu_t &= \frac{1}{q}\sum_{j=1}^{q} \mu_{t-j} + \eta_{2,t}\ . \end{aligned} \tag{8}$$

Here $q \geq 1$ controls the extent of smoothness. The higher the q, the smoother the level $\{\mu_1, \mu_2, ..., \mu_n\}$. We experiment with $q = 1, 3, 7, 14$.

Again, we cast the model into an SSM. The dimension of the state vector is $m = d - 1 + q$.

$$\alpha_t = \begin{pmatrix} i_t \\ \vdots \\ i_{t-d+2} \\ \mu_t \\ \vdots \\ \mu_{t-q+1} \end{pmatrix}, \quad \eta_t = \begin{pmatrix} \eta_{1,t} \\ \eta_{2,t} \end{pmatrix}.$$

We describe **Z**, **R**, **T** by the sparse matrix format. Denote the (i, j)th element of a matrix, for example, **T**, by $\mathbf{T}(i, j)$ (one index for vectors). An element is zero unless specified.

$$\begin{aligned}
&\mathbf{Z} = [\mathbf{Z}(i,j)]_{1\times m}, && \mathbf{Z}(1) = 1,\ \mathbf{Z}(d) = 1,\\
&\mathbf{R} = [\mathbf{R}(i,j)]_{m\times 2}, && \mathbf{R}(1,1) = 1,\ \mathbf{R}(d,2) = 1,\\
&\mathbf{T} = [\mathbf{T}(i,j)]_{m\times m}, && \mathbf{T}(1,j) = -1,\ j = 1,2,...,d-1,\\
& && \mathbf{T}(1+j,j) = 1,\ j = 1,2,...,d-2,\\
& && \mathbf{T}(d,d-1+j) = \frac{1}{q},\ j = 1,...,q,\\
& && \mathbf{T}(d+j,d-1+j) = 1, j = 1,2,...,q-1, \text{ if } q > 1.
\end{aligned}$$

We compare the LS and GLS models in Section 5 by experiments. It is shown that for distant prediction, imposing smoothness on the level can improve performance.

In practice, the prediction of a future x_{t+h} based on $\{x_1, x_2, ..., x_t\}$ comprises two steps:

1. Estimate **H** and **Q** in GLS (or SSM in general) using the past series $\{x_1, ..., x_t\}$.

2. Estimate x_{t+h} by the conditional expectation $E(x_{t+h} \mid x_1, x_2, ..., x_t)$ under the estimated model.

We may not need to re-estimate the model with every new coming x_t, but update the model once every batch of data. We estimate the model by the ML criterion using the EM algorithm. The Kalman filter and smoother, which involve forward and backward recursion respectively, are the core of the EM algorithm for SSM. Given an estimated model, the Kalman filter is used again to compute $E(x_{t+h} \mid x_1, x_2, ..., x_t)$, as well as the variance $Var(x_{t+h} \mid x_1, x_2, ..., x_t)$: a useful indication for the prediction accuracy. Details on the algorithms for estimating SSM and making prediction based on SSM are provided in the Appendix. A thorough coverage on the theories of SSM and related computational methods is referred to Durbin and Koopman (2001).

Because treating impulses improves prediction, as demonstrated by the experiments in Section 5, it is conducted for the GLS approach. In particular, we invoke the impulse detection embedded in HW. For any segment of time where an impulse is marked, the observed data x_t are replaced by $L_t + I_t$ computed by HW. This modified series is then input to the GLS estimation and prediction algorithms. The schematic diagram for forecasting using GLS is shown in Figure 2(b).

4. Long-range Trend and Seasonality

Web page views sometimes show long-range trend and seasonality. In Figure 7(a), three time series over a period of four years are shown. Detailed description of the series is provided in Section 5. Each time series demonstrates apparently a global trend and yearly seasonality. For instance, the first series, namely `amazon`, grows in general over the years and peaks sharply every year around December. Such long-range patterns can be exploited for forecasting, especially for distant future. To effectively extract long-range trend and season, several needs ought to be addressed:

1. Assume the period of the long-range season is a year. Because the Internet is highly dynamic, it is necessary to derive the yearly season using past data over recent periods and usually only a few (e.g., two) are available.

2. A mechanism to control the smoothness of the long-range season is needed. By enforcing smoothness, the extracted season tends to be more robust, a valuable feature especially when given limited past data.

3. The magnitude of the yearly season may vary across the years. As shown in Figure 7(a), although the series over different years show similar patterns, the patterns may be amplified or shrunk over time. The yearly season thus should be allowed to scale.

The HW and GLS approaches fall short of meeting the above requirements. They exploit mainly the local statistical dependence in the time series. Because HW (and similarly GLS) performs essentially exponential smoothing on the level and linear growth terms, the effect of historic data further away attenuates fast. HW is not designed to extract a global trend over multiple years. Furthermore, HW requires a relatively large number of periods to settle to the intended season; and importantly, HW assumes a fixed season over the years. Although HW is capable of adjusting with a slowly changing season when given enough periods of data, it does not directly treat the scaling of the season, and hence is vulnerable to the scaling phenomenon.

In our study, we adopt a linear regression approach to extract the long-range trend. We inject elasticity into the yearly season and allow it to scale from a certain yearly pattern. The algorithm developed is called *Elastic Smooth Season Fitting (ESSF)*. The time unit of the series is supposed to be a day.

4.1 Elastic Smooth Season Fitting

Before extracting long-range trend and season, we apply HW with impulse detection to obtain the weekly season and the smoothed level series L_t, $t = 1,...,n$. Recall that the HW prediction for the level L_{t+h} at time t is L_t, assuming no linear growth term in our experiments. We want to exploit the global trend and yearly season existing in the level series to better predict L_{t+h} based on $\{L_1, L_2, ..., L_t\}$.

We decompose the level series L_t, $t = 1,...,n$, into a yearly season, y_t, a global linear trend u_t, and a volatility part n_t:

$$L_t = u_t + y_t + n_t \ , \quad t = 1,2,...,n \ .$$

Thus the original series x_t is decomposed into:

$$x_t = u_t + y_t + n_t + I_t + N_t \ , \tag{9}$$

where I_t and N_t are the season and noise terms from HW. Let $\mathbf{u} = \{u_1, u_2, ..., u_n\}$ and $\mathbf{y} = \{y_1, y_2, ..., y_n\}$. They are solved by the following iterative procedure. At this moment, we assume the ESSF algorithm, to be described shortly, is available. We start by setting $\mathbf{y}^{(0)} = 0$. At iteration p, update $\mathbf{y}^{(p)}$ and $\mathbf{u}^{(p)}$ by

1. Let $g_t = L_t - y_t^{(p-1)}$, $t = 1,...,n$. Note g_t is the global trend combined with noise, taking out the current additive estimate of the yearly season.

2. Perform linear regression of $\mathbf{g} = \{g_1, ..., g_n\}$ on the time grid $\{1,2,...,n\}$. Let the regressed value at t be $u_t^{(p)}$, $t = 1,2,...,n$. Thus for some scalars $b_1^{(p)}$ and $b_2^{(p)}$, $u_t^{(p)} = b_1^{(p)} t + b_2^{(p)}$.

3. Let $z_t = L_t - u_t^{(p)}$, $t = 1, ..., n$. Here z_t is the yearly season combined with noise, taking out the current estimate of the global trend. Apply ESSF to $\mathbf{z} = \{z_1, z_2, ..., z_n\}$. Let the yearly season derived by ESSF be $\mathbf{y}^{(p)}$.

It is analytically difficult to prove the convergence of the above procedure. Experiments based on three series show that the difference in $\mathbf{y}^{(p)}$ reduces very fast. At iteration p, $p \geq 2$, we measure the relative change from $\mathbf{y}^{(p-1)}$ to $\mathbf{y}^{(p)}$ by

$$\frac{||\mathbf{y}^{(p)} - \mathbf{y}^{(p-1)}||}{||\mathbf{y}^{(p-1)}||}, \tag{10}$$

where $||\cdot||$ is the L_2 norm. Detailed results are provided in Section 5. Because ESSF always has to be coupled with global trend extraction, for brevity, we also refer to the entire procedure above as ESSF when the context is clear, particularly, in Section 4.2 and Section 5.

We now present the ESSF algorithm for computing the yearly season based on the trend removed $\mathbf{z}$. For notational brevity, we re-index day t by double indices (k, j), which indicates day t is the jth day in the kth year. Denote the residue $z_t = L_t - u_t$ by $z_{k,j}$, the yearly season y_t by $y_{k,j}$ (we abuse the notation here and assume the meaning is clear from the context), and the noise term n_t by $n_{k,j}$. Suppose there are a total of K years and each contains D days. Because leap years contain one more day, we take out the extra day from the series before applying the algorithm.

We call the yearly season pattern $\bar{\mathbf{y}} = \{\bar{y}_1, \bar{y}_2, ..., \bar{y}_D\}$ the *season template*. Since we allow the yearly season $y_{k,j}$ to scale over time, it relates to the season template by

$$y_{k,j} = \alpha_{k,j}\bar{y}_j, \quad k = 1, 2, ..., K, \ j = 1, ..., D,$$

where $\alpha_{k,j}$ is the scaling factor. One choice for $\alpha_{k,j}$ is to let $\alpha_{k,j} = c_k$, that is, a constant within any given year. We call this scheme step-wise constant scaling since $\alpha_{k,j}$ is a step function if single indexed by time t. One issue with the step-wise constant scaling factor is that $y_{k,j}$ inevitably jumps when entering a new year. To alleviate the problem, we instead use a piece-wise linear function for $\alpha_{k,j}$. Let $c_0 = 1$. Then

$$\alpha_{k,j} = \frac{j-1}{D}c_k + \frac{D-j+1}{D}c_{k-1}, \quad k = 1, 2, ..., K, \ j = 1, ..., D. \tag{11}$$

The number of scaling factors c_k to be determined is still K. Let $\mathbf{c} = \{c_1, ..., c_K\}$. At the first day of each year, $\alpha_{k,1} = c_{k-1}$. We optimize over both the season template $\bar{y}_j$, $j = 1, ..., D$, and the scaling factors c_k, $k = 1, ..., K$.

We now have

$$z_{k,j} = \alpha_{k,j}\bar{y}_j + n_{k,j},$$

where $z_{k,j}$'s are given, while c_k, $\bar{y}_j$, and $n_{k,j}$, $k = 1, ..., K$, $j = 1, ..., D$, are to be solved. A natural optimization criterion is to minimize the sum of squared residues:

$$\min_{\bar{\mathbf{y}}, \mathbf{c}} \sum_k \sum_j n_{k,j}^2 = \min_{\bar{\mathbf{y}}, \mathbf{c}} \sum_k \sum_j (z_{k,j} - \alpha_{k,j}\bar{y}_j)^2 .$$

If the number of years K is small, $\bar{\mathbf{y}}$ obtained by the above optimization can be too wiggly. We add a penalty term to ensure the smoothness of $\bar{\mathbf{y}}$. The discrete version of the second order derivative for $\bar{y}_j$ is

$$\ddot{\bar{y}}_j = \bar{y}_{j+1} + \bar{y}_{j-1} - 2\bar{y}_j ,$$

and $\sum_j \ddot{\bar{y}}_j^2$ is used as the smoothness penalty. Since $\bar{\mathbf{y}}$ is one period of the yearly season, when j' is out of the range $[1,D]$, $\bar{y}_{j'}$ is understood as $\bar{y}_{j' \bmod D}$. For instance, $\bar{y}_0 = \bar{y}_D$, $\bar{y}_{D+1} = \bar{y}_1$. We form the following optimization criterion with a pre-selected regularization parameter λ:

$$\min_{\bar{\mathbf{y}},\mathbf{c}} G(\bar{\mathbf{y}},\mathbf{c}) = \min_{\bar{\mathbf{y}},\mathbf{c}} \sum_k \sum_j (z_{k,j} - \alpha_{k,j}\bar{y}_j)^2 + \lambda \sum_j (\bar{y}_{j+1} + \bar{y}_{j-1} - 2\bar{y}_j)^2 \,. \tag{12}$$

To solve (12), we alternate the optimization of $\bar{\mathbf{y}}$ and $\mathbf{c}$. With either fixed, $G(\bar{\mathbf{y}},\mathbf{c})$ is a convex quadratic function. Hence a unique minimum exists and can be solved by a multivariable linear equation. The algorithm is presented in details in Appendix B.

Experiments show that allowing scalable yearly season improves prediction accuracy, so does the smoothness regularization of the yearly season. As long as λ is not too small, the prediction performance varies marginally for a wide range of values. The sensitivity of prediction accuracy to λ is studied in Section 5.

A more ad-hoc approach to enforce smoothness is to apply moving average to the yearly season extracted without smoothness regularization. We can further simplify the optimization criterion in (12) by employing step-wise constant scaling factor, that is, let $\alpha_{k,j} = c_k$, $k = 1,...,K$. The jump effect caused by the abrupt change of the scaling factor is reduced by the moving average as well. Specifically, the optimization criterion becomes

$$\min_{\bar{\mathbf{y}},\mathbf{c}} \tilde{G}(\bar{\mathbf{y}},\mathbf{c}) = \min_{\bar{\mathbf{y}},\mathbf{c}} \sum_k \sum_j (z_{k,j} - c_k\bar{y}_j)^2 \,. \tag{13}$$

The above minimization is solved again by alternating the optimization of $\bar{\mathbf{y}}$ and $\mathbf{c}$. See Appendix B for details. Comparing with Eq. (12), the optimization for (13) reduces computation significantly. After acquiring $\bar{\mathbf{y}}$, we apply a double sided moving average. We call the optimization algorithm for (13) combined with the post operation of moving average the fast version of ESSF. Experiments in Section 5 show that ESSF Fast performs similarly to ESSF.

4.2 Prediction

We note again that ESSF is for better prediction of the level L_t obtained by HW. To predict x_t, the weekly season extracted by HW should be added to the level L_t. The complete process of prediction is summarized below. We assume that prediction starts on the 3rd year since the first two years have to serve as past data for computing the yearly season.

1. Apply HW to obtain the weekly season I_t, and the level L_t, $t = 1,2,...,n$.

2. At the beginning of each year k, $k = 3,4,...$, take the series of L_t's in the past two years (year $k-2$ and $k-1$) and apply ESSF to this series to solve the yearly season template $\bar{\mathbf{y}}$ and the scaling factors, c_1 and c_2 for year $k-2$ and $k-1$ respectively. Predict the yearly season for future years $k' \geq k$ by $c_2\bar{\mathbf{y}}$. Denote the predicted yearly season at time t in any year $k' \geq k$ by $Y_{t,k}$, where the second subscript clarifies that only the series before year k is used by ESSF.

3. Denote the year in which day t lies by $\nu(t)$. Let the yearly season removed level be $\tilde{L}_t = L_t - Y_{t,\nu(t)}$. At every t, apply linear regression to $\{\tilde{L}_{t-2D+1},...,\tilde{L}_t\}$ over the time grid $\{1,2,...,2D\}$. The slope of the regressed line is taken as the long-range growth term $\tilde{T}_t$.

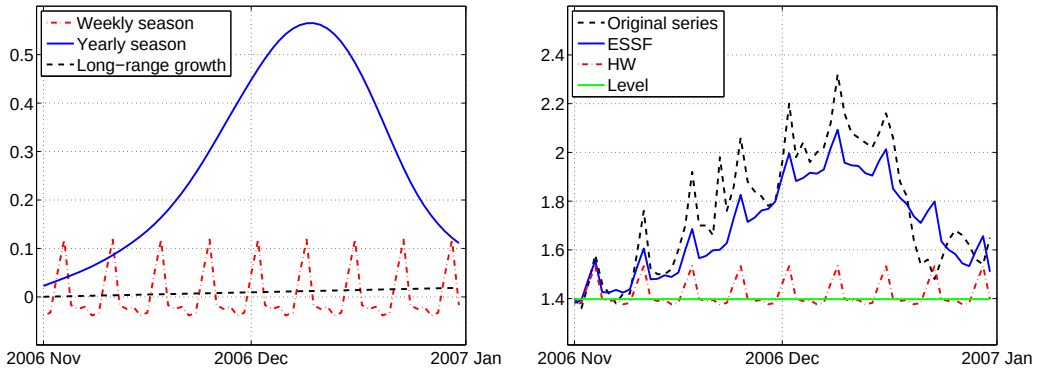


Figure 3: Decomposition of the prediction terms for the amazon series in November and December of 2006 based on data up to October 31, 2006: (a) The weekly season, yearly season, and long-range linear growth terms in the prediction; (b) Comparison of the predicted series by HW and ESSF.

Suppose at the end of day t (or beginning of day $t+1$), we predict for the hth day ahead of t. Let the prediction be $\hat{x}_{t+h}$. Also let $r(t+h)$ be the smallest integer such that $t+h-r(t+h)\cdot d \le t$ (d is the weekly period). Then,

$$\hat{x}_{t+h} = \tilde{L}_t + h\tilde{T}_t + Y_{t+h,\nu(t)} + I_{t+h-r(t+h)\cdot d}\ . \tag{14}$$

Drawing a comparison between Eqs. (14) and (9), we see that $\tilde{L}_t + h\tilde{T}_t$ is essentially the prediction for the global linear trend term u_{t+h}, $Y_{t+h,\nu(t)}$ the prediction for the yearly season y_{t+h}, and $I_{t+h-r(t+h)\cdot d}$ the prediction for the weekly season I_{t+h}. The schematic diagram for forecasting by ESSF is shown in Figure 2(c).

If day $t+h$ is in the same year as t, $Y_{t+h,\nu(t)} = Y_{t+h,\nu(t+h)}$ is the freshest possible prediction for the yearly season at $t+h$. If instead $\nu(t) < \nu(t+h)$, the yearly season at $t+h$ is predicted based on data more than one year ago. One might have noticed that we use only two years of data to extract the yearly season regardless of the available amount of past data. This is purely an individual choice due to our preference of using recent data. Experiments based on the series described in Section 5 show that whether all the available past data are used by ESSF causes negligible difference in prediction performance.

To illustrate the roles of the terms in the prediction formula (14), we plot them separately in Figure 3(a) for the amazon series. The series up to October 31, 2006 is assumed to have been observed, and the prediction is for November and December of 2006. Figure 3(a) shows that during these two months, the predicted yearly season is much more prominent than the weekly season and the slight linear growth. Figure 3(b) compares the prediction by ESSF and HW respectively. The series predicted by HW is weekly periodic with a flat level, while that by ESSF incorporates the yearly seasonal variation and is much closer to the original series, as one might have expected.

5. Experiments

We conduct experiments using twenty six time series. As a study of the characteristics of Web page views, we examine the significance of the seasonal as well as impulsive variations. Three relatively short series are used to assess the performance of short-term prediction by the HW and GLS approaches. The other twenty three series are used to test the ESSF algorithm for long-term prediction. In addition to comparing the different forecasting methods, we also present results to validate the algorithmic choices made in ESSF.

5.1 Data Sets

We conduct experiments based on the time series described below.

1. The `Auton` series records the daily page views of the Auton Lab, headed by Andrew Moore, in the Robotics Institute at the Carnegie Mellon University (*http://www.autonlab.org*). This series spans from August 14, 2005 to May 1, 2007, a total of 626 days.

2. The `Wang` series records the daily page views of the Web site for the research group headed by James Wang at the Pennsylvania State University (*http://wang.ist.psu.edu*). This series spans from January 1, 2006 to February 1, 2008, a total of 762 days.

3. The `citeseer` series records the hourly page views to citeseer, an academic literature search engine currently located at *http://citeseer.ist.psu.edu*. This series spans from 19 : 00 on September 6, 2005 to 4 : 00 on September 25, 2005, a total of 442 hours.

4. We acquired 23 relatively long time series from the site *http://www.google.com/trends*. This Web site provides search volumes for user specified phrases. We treat the search volumes as an indication of the page views to dynamically generated Web pages by Google. The series record daily volumes from Jan, 2004 to December 30, 2007 (roughly four full years), a total of 1460 days. The volumes for each phrase are normalized with respect to the average daily volume of that phrase in the month of January 2004. The normalization will not affect the prediction accuracy, which is measured relatively with respect to the average level of the series. We also call the series collectively the `g-trends` series.

5.2 Evaluation

Let the prediction for x_t be $\hat{x}_t$. Suppose prediction is provided for a segment of the series, $\{x_{t_0+1}, x_{t_0+2}, \ldots, x_{t_0+J}\}$, where $0 \le t_0 < n$. We measure the prediction accuracy by the error rate defined as

$$R_e = \sqrt{\frac{RSS}{SSS}}$$

where *RSS*, the *residual sum of squares* is

$$RSS = \sum_{t=t_0+1}^{t_0+J} (\hat{x}_t - x_t)^2 \tag{15}$$

and *SSS*, the *series sum of squares* is

$$SSS = \sum_{t=t_0+1}^{t_0+J} x_t^2 \,. \tag{16}$$

We call R_e the prediction error rate. It is the reciprocal of the square root of the signal to noise ratio (SNR), a measure commonly used in signal processing. We can also evaluate the effectiveness of a prediction method by comparing *RSS* to *SPV*, the *sum of predictive variation*:

$$SPV = \sum_{t=t_0+1}^{t_0+J} (x_t - \bar{x}_t)^2, \quad \bar{x}_t = \frac{\sum_{\tau=1}^{t-1} x_\tau}{t-1}.$$

We can consider $\bar{x}_t$, the mean up to time $t-1$, as the simplest prediction of x_t using past data. We call this scheme of prediction Mean of Past (MP). *SPV* is essentially the *RSS* corresponding to the MP method. The R_e of MP is $\sqrt{SPV/SSS}$. We denote the ratio between the R_e of a prediction method and that of MP by $Q_e = \sqrt{RSS/SPV}$, referred to as the error ratio. As a measure on the amount of error, in practice, R_e is more pertinent than Q_e for users concerned with employing the prediction in subsequent tasks. We thus use R_e as the major performance measure in all the experimental results. For comparison with baseline prediction methods, we also show R_e of MP as well as that of the Moving Average (MA). In the MA approach, considering the weekly seasonality, we treat Monday to Sunday separately. Specifically, if a day to be predicted is a Monday, we forecast by the average of the series on the past 4 Mondays. Similarly for the other days of a week. For the hourly page view series with daily seasonality, MA predicts by the mean of the same hours in the past 4 days.

As discussed previously, Web page views exhibit impulsive changes. The prediction error during an impulse is extraordinarily large, skewing the average error rate significantly even if impulses only exist on a small fraction of the series. The bias caused by the outlier errors is especially strong when the usual amount of page views is low. We reduce the effect of outliers by removing a small percentage of large errors, in particular, 5% in our experiments. Without loss of generality, suppose the largest (in magnitude) 5% errors are $\hat{x}_t - x_t$ at $t_0 + 1 \le t \le t_1$. We adjust RSS and SSS by using only $\hat{x}_t - x_t$ at $t > t_1$ and compute the corresponding R_e. Specifically,

$$RSS_{adj} = \sum_{t=t_1+1}^{t_0+J} (\hat{x}_t - x_t)^2, \; SSS_{adj} = \sum_{t=t_1+1}^{t_0+J} x_t^2, \; R_e^{adj} = \sqrt{\frac{RSS_{adj}}{SSS_{adj}}}.$$

We report both R_e and R_e^{adj} to measure the prediction accuracy for the series `Auton`, `Wang`, and `citeseer`. For the twenty three `g-trends` series, because there is no clear impulse, we use only R_e.

Because the beginning portion of the series with a certain length is needed for initialization in HW, SSM, or ESSF, we usually start prediction after observing $t_0 > 0$ time units. Moreover, we may predict several time units ahead for the sum of the series over a run of multiple units. The ground truth at t is not necessarily x_t. In general, suppose prediction starts after t_0 and is always for a stretch of w time units that starts h time units ahead. We call w the *window size* of prediction and h the *unit ahead*.

Let the whole series be $\{x_1, x_2, ..., x_n\}$. In the special case when $h = 1$ and $w = 1$, after observing the series up to $t-1$, we predict for x_t, $t = t_0 + 1, ..., n$. The ground truth at t is x_t. If $h \ge 1$, $w = 1$, we predict for x_t, $t = t_0 + h, ..., n$, after observing the series up to $t - h$. Let the predicted value be $\hat{x}_{t,-h}$, where the subscript $-h$ emphasizes that only data h time units ago are used. If $h \ge 1$, $w \ge 1$, we predict for

$$\tilde{x}_t = \sum_{\tau=t}^{t+w-1} x_\tau, \quad t = t_0 + h, ..., n - w + 1,$$

after observing the series up to $t-h$. The predicted value at t is

$$\hat{\tilde{x}}_t = \sum_{\tau=t}^{t+w-1} \hat{x}_{\tau,-h}\,.$$

To compute the error rate R_e, we adjust *RSS* and *SSS* in Eq. (15) and (16) according to the ground truth:

$$RSS = \sum_{t=t_0+h}^{n-w+1} (\hat{\tilde{x}}_t - \tilde{x}_t)^2\,, \quad SSS = \sum_{t=t_0+h}^{n-w+1} \tilde{x}_t^2\,.$$

For the series `Auton`, `Wang`, and `citeseer`, $t_0 = 4d$, where d is the season period. The segment $\{x_1,...,x_{4d}\}$ is used for initialization by both HW and SSM. For the `g-trends` series, $t_0 = 731$. That is, the first two years of data are used for initialization. Two years of past data are needed because the ESSF algorithm requires at least two years of data to operate.

5.3 Results

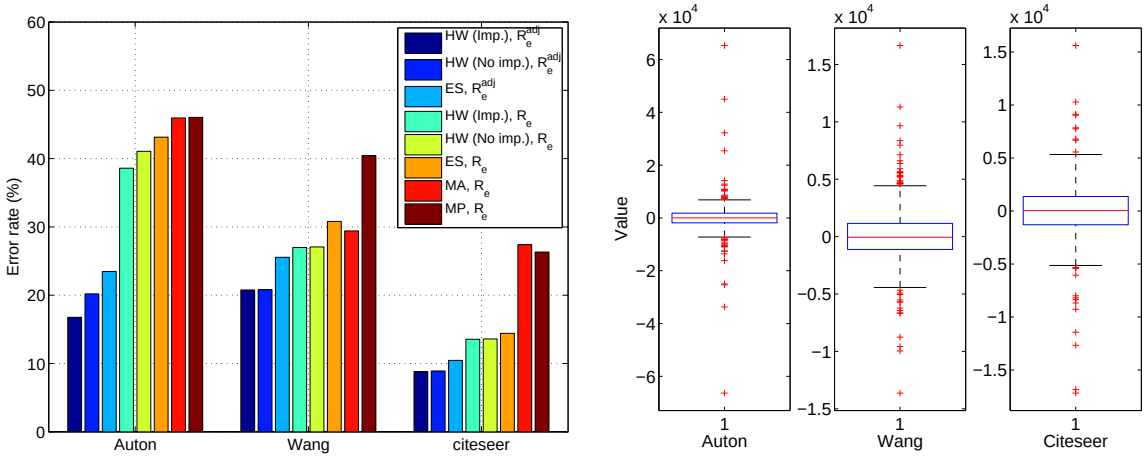
For the series `Auton`, `Wang`, and `citeseer`, we focus on short-term prediction no greater than 30 time units ahead. Because the series are not long enough for extracting long-range trend and season by the ESSF algorithm, we only test the HW procedure with or without impulse detection and the GLS approach. For the twenty three `g-trends` series, we compare ESSF with HW for prediction up to half a year ahead.

5.3.1 SHORT-TERM PREDICTION

Web page views often demonstrate seasonal variation, sometimes at multiple scales. The HW procedure given by Eq. (2)∼(4) and the GLS model specified in Eq. (8) both assume a season term with period d. In our experiments, for the daily page view series `Auton` and `Wang`, $d = 7$ (a week), while for the hourly series `citeseer`, $d = 24$ (a day). As mentioned previously, the local linear growth term in Eq. (3) is removed in our experiments because it is found not helpful. The smoothing parameters for the level and the season terms in Eq. (2) and (4) are set to $\zeta = 0.5$ and $\delta = 0.25$. Because HW has no embedded mechanism to select these parameters, we do not aggressively tune them and use the same values for all the experiments reported here.

To assess the importance of weekly (or daily) seasonality for forecasting, we compare HW and and its reduced form without the season term. Similarly as the linear growth term, the season term can be deleted by initializing it to zero and setting its corresponding smoothing parameter δ in Eq. (4) to zero. The reduced HW procedure without the local linear growth and season terms is essentially Exponential Smoothing (ES) (Chatfield, 2004). Figure 4(a) compares the prediction performance in terms of R_e and R_e^{adj} for the three series by HW and ES. Results for two versions of HW, with and without treating impulses, are provided. The comparison of the two versions will be discussed shortly. Results obtained from the two baseline methods MA and MP are also shown. For each of the three series, HW (both versions), which models seasonality, consistently outperforms ES, reflecting the significance of seasonality in these series. We also note that for the `auton` series, R_e is almost twice as large as R_e^{adj} although only 5% of outliers are removed. This dramatic skew of the error rate is caused by the short but strong impulses occurred in this series.

To evaluate the impulse-resistant measure, described in Section 2, we compare HW with and without impulse detection in Figure 4(a). Substantial improvement is achieved for the `Auton` series.



(a) (b)

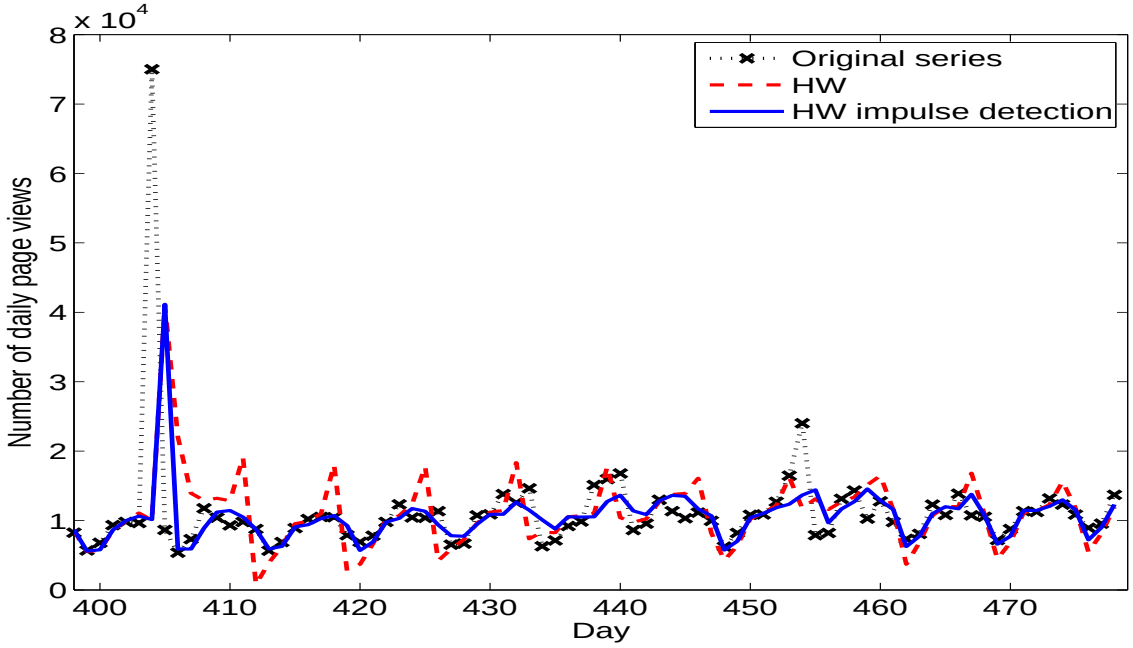
Figure 4: Compare the prediction performance in terms of R_e^{adj} and R_e on the three series `Auton`, `Wang`, and `citeseer` using different methods: HW with or without impulse detection, ES without season, MA, and MP. (a) The error rates. (b) The box plots for the differences of page views at adjacent time units.

The decrease in error rate for the other two series is small, a result of the fact there is no strong impulse in them. To directly demonstrate the magnitude of the impulses, we compute the differences in page view between adjacent time units, $\{x_2 - x_1, x_3 - x_2, ..., x_n - x_{n-1}\}$, and show the box plots for their distributions in Figure 4(b). The stronger impulses in `Auton` are evident from the box plots. Comparing with the other two series, the middle half of the `Auton` data (between the first and third quartiles), indicated by the box in the plot, is much narrower relative to the overall range of the data. In another word, the outliers deviate more severely from the majority mass of the data.

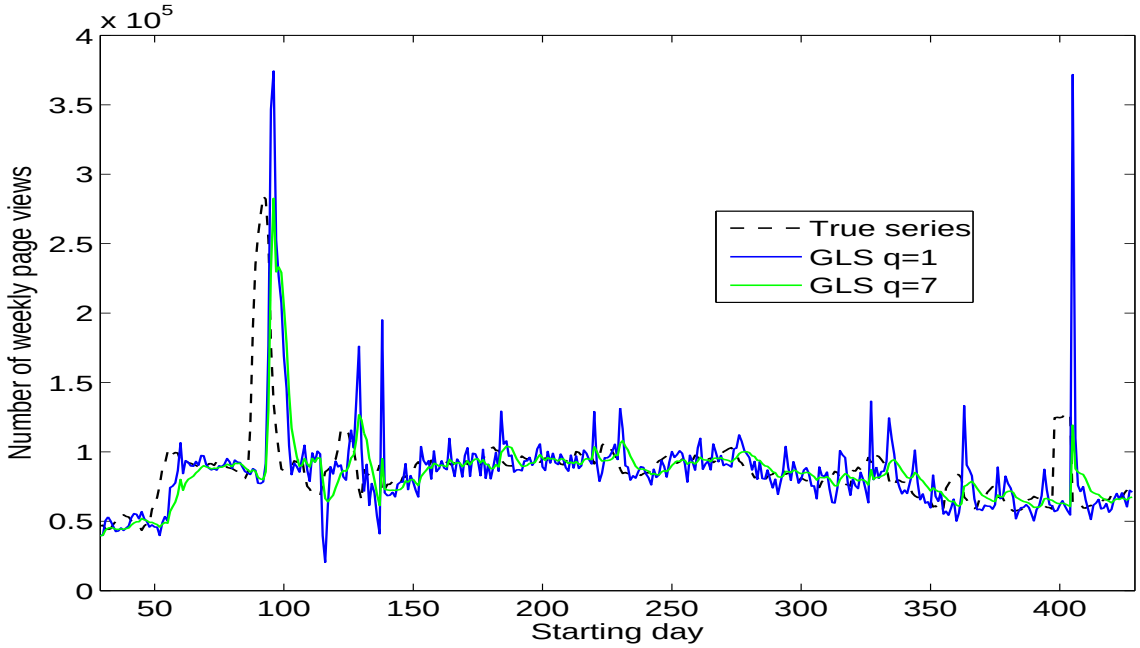
To illustrate the gain from treating impulses, we also show the predicted series for `Auton` in Figure 5(a). For clarity of the plot, only a segment of the series around an impulse is shown. The predicted series by HW with impulse detection returns close to the original series shortly after the impulse, while that without ripples with large errors over several periods afterward. In the sequel, for both HW and GLS, impulse detection is included by default.

Table 1 lists the error rates for the three series using different methods and under different pairs of (h,w), where h is the unit ahead and w is the window size of prediction. We provide the error rate R_e^{adj} in addition to R_e to show the performance on impulse excluded portion of the series. For `Auton` and `Wang`, $(h,w) = (1,1),(1,7),(1,28)$. For `citeseer`, $(h,w) = (1,1),(1,12),(1,24)$. For the GLS model, a range of values for the smooth parameter q are tested. As shown by the table, when $(h,w) = (1,1)$, the performance of HW and that of GLS at the best q are close. When predicting multiple time units, for example, $w = 7,28$ or $w = 12,24$, GLS with $q > 1$ achieves better accuracy. For `Wang` and `citeseer`, at every increased w, the lowest error rates are obtained by GLS with an increased q. This supports the heuristic that when predicting for a more distant time, smoother prediction is preferred to reduce the influence of local fluctuations.

We compare the predicted series for `Auton` by GLS with $q = 1$ and $q = 7$ in Figure 5(b). Here, the unit ahead $h = 1$, and the window size $w = 7$. The fluctuation of the predicted series obtained



(a)



(b)

Figure 5: Compare predicted series for `Auton`: (a) Results obtained by HW with and without impulse detection. The unit ahead h and window size w of prediction are 1; (b) Results obtained by GLS with $q = 1$ and $q = 7$. The unit ahead is 1, and window size is 7 (a week).

Error rate R_e (%)	HW	GLS q=1	GLS q=3	GLS q=7	GLS q=14
Auton: 1 Day	**38.60**	41.46	40.05	41.10	41.74
7 Days	34.52	36.78	34.70	30.33	**28.41**
28 Days	32.34	34.63	32.60	25.80	**21.93**
Wang: 1 Day	26.99	**26.19**	26.24	26.51	26.77
7 Days	19.95	16.19	**16.09**	16.27	16.48
28 Days	21.11	16.44	16.30	16.31	**16.05**
citeseer: 1 Hour	13.55	**13.18**	14.01	15.00	16.29
12 Hours	15.04	14.63	13.66	**12.96**	13.10
24 Hours	15.47	15.87	14.88	14.00	**13.80**

Error rate R_e^{adj} (%)	HW	GLS q=1	GLS q=3	GLS q=7	GLS q=14
Auton: 1 Day	16.76	18.02	17.45	**16.53**	18.14
7 Days	15.60	15.63	14.55	**12.94**	13.45
28 Days	17.32	17.49	16.68	**15.03**	15.31
Wang: 1 Day	20.77	**20.41**	20.64	20.98	20.99
7 Days	16.29	13.52	**13.38**	13.44	13.55
28 Days	16.83	13.65	13.49	13.45	**13.26**
citeseer: 1 Hour	8.80	**8.17**	8.95	10.38	12.16
12 Hours	12.53	10.74	**10.70**	10.97	11.45
24 Hours	12.98	12.14	11.98	11.89	**11.82**

Table 1: The prediction error rates R_e and R_e^{adj} for the three series Auton, Wang, and citeseer obtained by several methods. The window size of prediction takes multiple values, while the unit ahead is always 1. HW and the GLS model with several values of q are compared.

by $q = 1$ is more volatile than that by $q = 7$. The volatility of the predicted series by $q = 7$ is much closer to that of the true series. As shown in Table 1, the error rate R_e^{adj} achieved by $q = 7$ is 12.94%, while that by $q = 1$ is 15.63%.

Based on the GLS model, the variance of x_t conditioned on the past $\{x_1, ..., x_{t-1}\}$ can be computed. The equations for the conditional mean $E(x_t \mid x_1, ..., x_{t-1})$ (i.e., the predicted value) and variance $Var(x_t \mid x_1, ..., x_{t-1})$ are given in (19). Since the conditional distribution of x_t is Gaussian, we can thus calculate a confidence band for the predicted series, which may be desired in certain applications to assess the potential deviation of the true values. Figure 6 shows the 95% confidence band for citeseer with $(h, w) = (1, 1)$. The confidence band covers nearly the entire original series.

GLS is more costly in computation than HW. We conduct the experiments using Matlab codes on 2.4GHz Dell computer with Linux OS. At $(h, w) = (1, 28)$ for Auton and Wang, and $(h, w) = (1, 24)$ for citeseer, the average user running time for sequential prediction along the whole series is respectively 0.51, 56.38, 65.87, 72.10, and 86.76 seconds for HW, and GLS at $q = 1, 3, 7, 14$. In our experiments, the GLS models are re-estimated after every $4d$ units, where d is the period. The computation in GLS is mainly spent on estimating the models and varies negligibly for different pairs of (h, w).

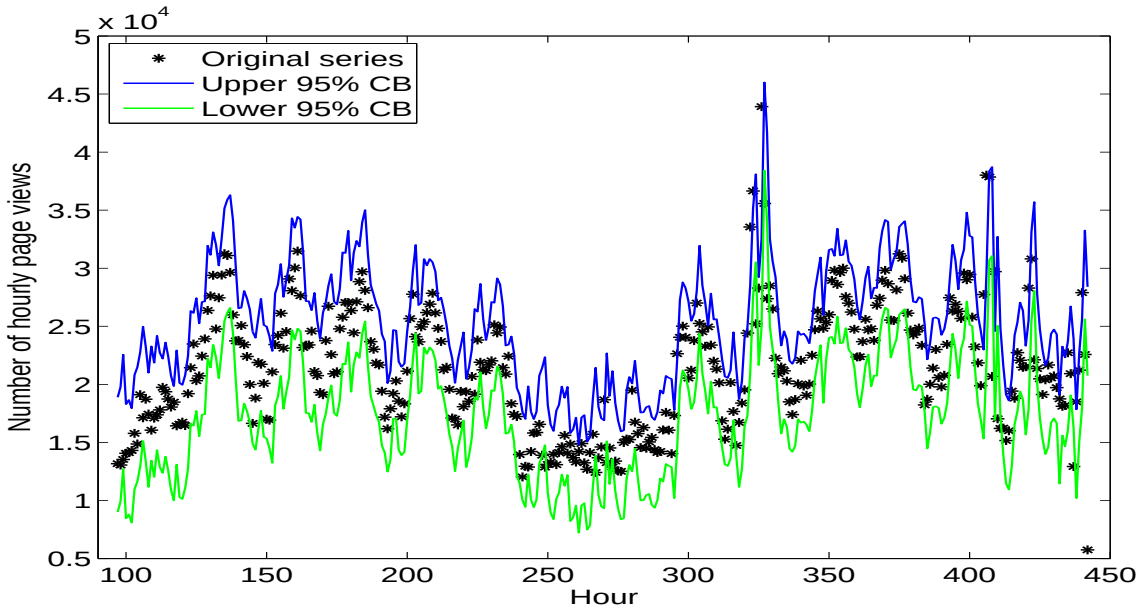


Figure 6: The predicted 95% confidence band for `citeseer` obtained by GLS with $q = 1$. The unit ahead h and window size w are 1.

5.3.2 LONG-TERM PREDICTION—A COMPREHENSIVE STUDY

We now examine the performance of ESSF based on the `g-trends` series. The first three series are acquired by the search phrases `amazon`, `Renoir` (French impressionism artist), and `greenhouse effect`, which will be used as the names for the series in the sequel. A comprehensive study with detailed results is first presented using these three series. Then, we expand the experiments to twenty additional `g-trends` series and present results on prediction accuracy and computational speed.

The original series of `amazon`, `Renoir`, and `greenhouse effect` averaged weekly are shown in Figure 7(a). Due to the weekly season, without averaging, the original series are too wiggly for clear presentation. Figure 7(c) and (d) show the yearly season templates extracted by ESSF from year 2004 and 2005 with smoothing parameter $\lambda = 0$, 1000 respectively. As expected, at $\lambda = 1000$, the yearly seasons are much smoother than those obtained at $\lambda = 0$, especially for the series `Renoir` and `greenhouse effect`. Figure 7(b) shows the scaling factors of the yearly seasons obtained by applying ESSF to the entire four years.

We compare the prediction obtained by ESSF with HW and the MA approach as a baseline. For ESSF, we test both $\lambda = 0$ and 1000, and its fast version with moving average window size 15. Prediction error rates are computed for the unit ahead h ranging from 1 to 180 days. We fix the prediction window size $w = 1$.

The error rates R_e obtained by the methods are compared in Figure 8(a), (b), (c) for `amazon`, `Renoir`, `greenhouse effect` respectively. Comparing with the other methods, the difference in the performance of HW and MA is marginal. When the unit ahead h is small, HW outperforms MA, but the advantage diminishes when h is large. For `Renoir` and `greenhouse effect`, HW becomes even inferior to MA when h is roughly above 60. ESSF with $\lambda = 1000$ and ESSF Fast perform

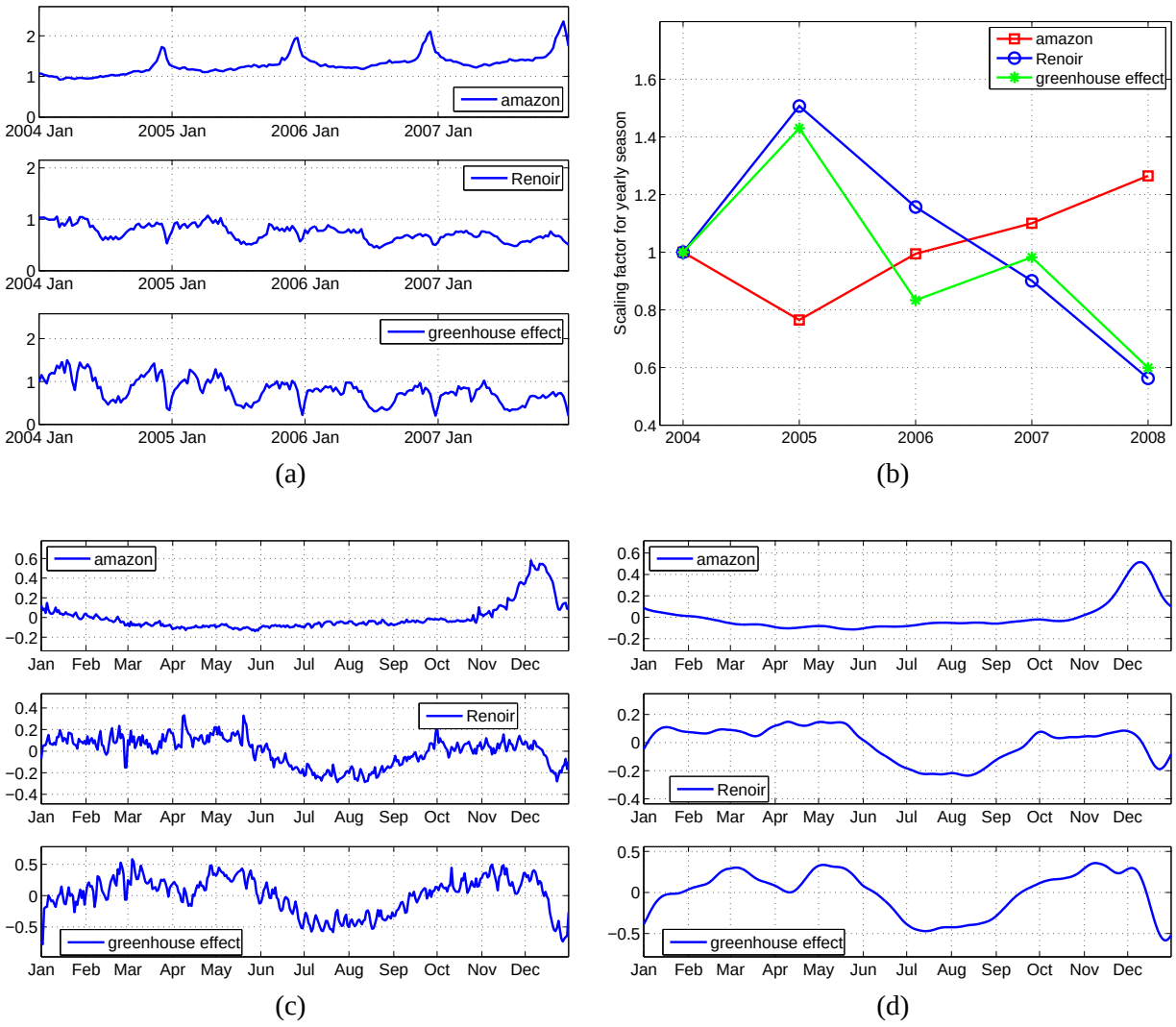


Figure 7: Extract the yearly seasons by ESSF for the `g-trends` series `amazon`, `Renoir`, and `greenhouse effect`: (a) The weekly averaged original series; (b) The scaling factor for the yearly season; (c) The yearly season extracted without smoothing at $\lambda = 0$; (d) The yearly season extracted with smoothing at $\lambda = 1000$.

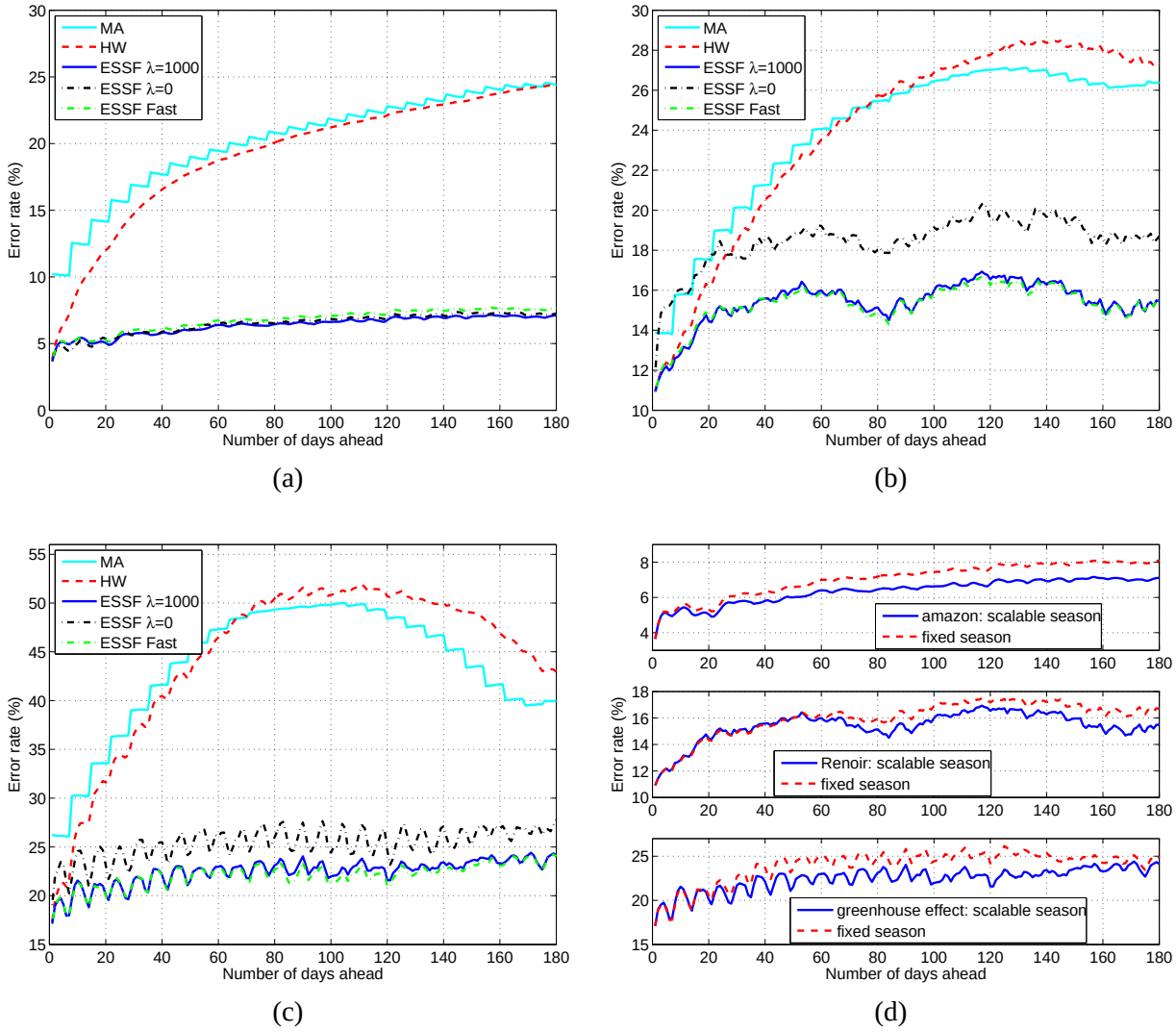
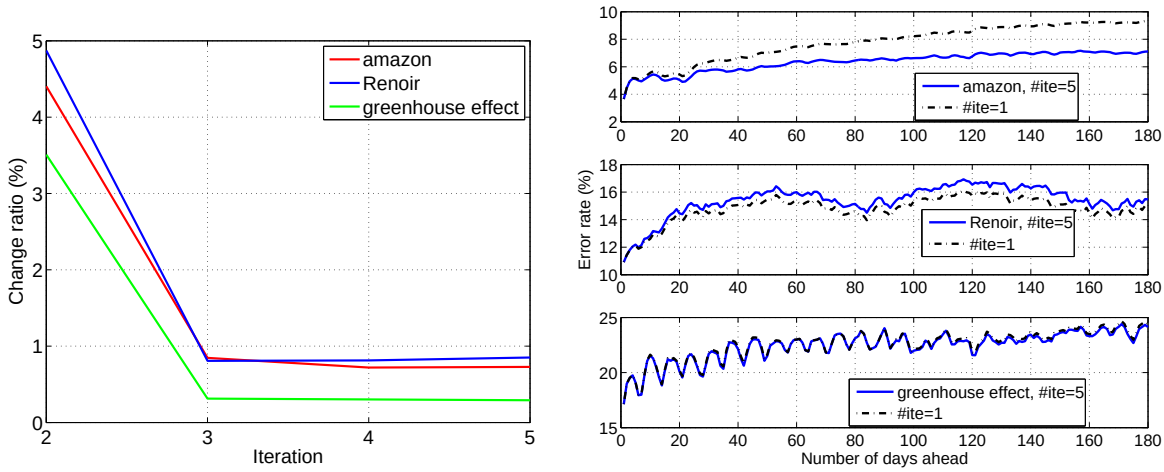


Figure 8: Compare prediction error rates R_e for three `g-trends` series using several methods. Prediction is performed for the unit ahead h ranging from 1 to 180, and a fixed the window size $w = 1$. Error rates obtained by MA, HW, ESSF with $\lambda = 1000$, 0, and the fast version of ESSF with moving average window size 15, are shown for the three series (a) `amazon`, (b) `Renoir`, (c) `greenhouse effect` respectively. The yearly season in ESSF is scalable. (d) Error rates obtained for the three series by ESSF, with $\lambda = 1000$, assuming a scalable yearly season versus fixed season.



(a) (b)

Figure 9: The effect of the number of iterations in the ESSF algorithm: (a) The change ratio in the extracted yearly season over the iterations; (b) Compare the error rates R_e obtained by ESSF with 1 iteration and 5 iterations respectively.

nearly the same, both achieving error rates consistently lower than those by HW. The gap between the error rates of ESSF and HW widens quickly with an increasing h. In general, when h increases, the prediction is harder, and hence the error rate tends to increase. The increase is substantially slower for ESSF than HW and MA. ESSF with $\lambda = 0$ performs considerably worse than $\lambda = 1000$ for `Renoir` and `greenhouse effect`, and closely for `amazon`. This demonstrates the advantage of imposing smoothness on the yearly season. We will study more thoroughly the effect of λ on prediction accuracy shortly.

Next, we experiment with ESSF under various setups and demonstrate the advantages of several algorithmic choices. First, recall that the fitting of the yearly season and the long-range trend is repeated multiple times, as described in Section 4.1. To study the effect of the number of iterations, we plot in Figure 9(a) the ratio of change in the yearly season after each iteration, as given by Eq. (10). For all the three series, the most prominent change occurs between iteration 1 and 2 and falls below 1% for any later iterations. We also compare the prediction error rates for $h = 1, \ldots, 180$ achieved by using only 1 iteration (essentially no iteration) versus 5 iterations. The results for the three series are plotted in Figure 9(b). The most obvious difference is with `amazon` for large h. At $h = 180$, the error rate obtained by 5 iterations is about 2% lower than by 1 iteration. On the other hand, even with only 1 iteration, the error rate at $h = 180$ is below 10%, much lower than the nearly 25% error rate obtained by HW or MA. For `greenhouse effect`, the difference is almost imperceptible.

In ESSF, the yearly season is not assumed simply as a periodic series. Instead, it can scale differently over the years based on the season template. To evaluate the gain, we compare ESSF with scalable yearly seasons versus fixed seasons. Here, the fixed season can be thought of as a special case of the scalable season with all the scaling parameters set to 1, or equivalently, the yearly season is the plain repeat of the season template. Figure 8(d) compares the error rates under

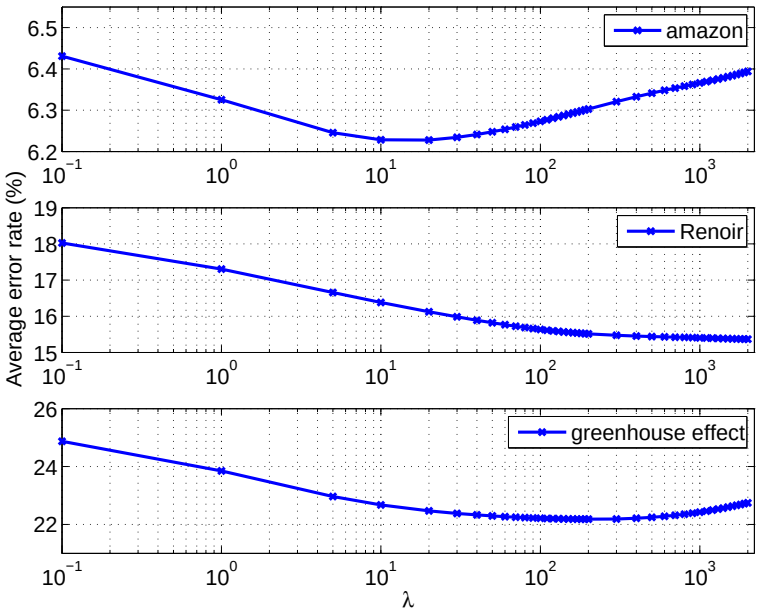


Figure 10: The effect of the smoothing parameter λ in ESSF on prediction accuracy. At each λ, the average of error rates R_e across the unit ahead $h = 1, ..., 180$ are shown.

the two schemes for the three series. Better performance is achieved by allowing scalable yearly seasons for all the three series. The advantage is more substantial when predicting the distant future.

To examine the sensitivity of the prediction accuracy to the smoothing parameter λ, we vary λ from 0.1 to 2000, and compute the error rates for $h = 1, ..., 180$. For concise illustration, we present the average of the error rates across h. Note that the results of $\lambda = 0, 1000$ at every h are shown in Figure 8, where $\lambda = 0$ is inferior. The variation of the average error rates with respect to λ (in log scale) is shown in Figure 10. For `amazon`, the error rates with different λ's lie in the narrow range of $[6.2\%, 6.45\%]$, while for `Renoir` and `greenhouse effect`, the range is wider, roughly $[15\%, 18\%]$ and $[22\%, 25\%]$ respectively. For all the three series, the decrease of the error rate is most steep when λ increases from 0.1 to 10. For $\lambda > 10$ and as large as 2000, the change in error rate is minor, indicating that the prediction performance is not sensitive to λ as long as it is not too small.

5.3.3 LONG-TERM PREDICTION—EXTENDED STUDY ON TWENTY TREND SERIES

We collect another twenty `g-trends` series with query phrases and corresponding series ID listed in Table 2. The error rates R_e achieved by the four methods: MA, HW, ESSF with $\lambda = 1000$, and ESSF Fast, over the twenty series are compared in Figure 11. The four plots in this figure each show results for predicting a single day in advance of h days, with $h = 1, 30, 60, 90$ respectively. For most series, MA is inferior to HW at every h. However, when h increases, the margin of HW over MA decreases. At $h = 1$, HW performs similarly as ESSF and ESSF Fast. At $h = 30, 60, 90$, both versions of ESSF, which achieve similar error rates between themselves, outperform HW.

To assess the predictability of the series, we compute the variation rates at $h = 1, 30, 60, 90$, shown in Figure 12(a). The variation rate at h is defined as $\sqrt{Var(x_{t+h} - x_t)} / \sqrt{Var(x_t)}$, where

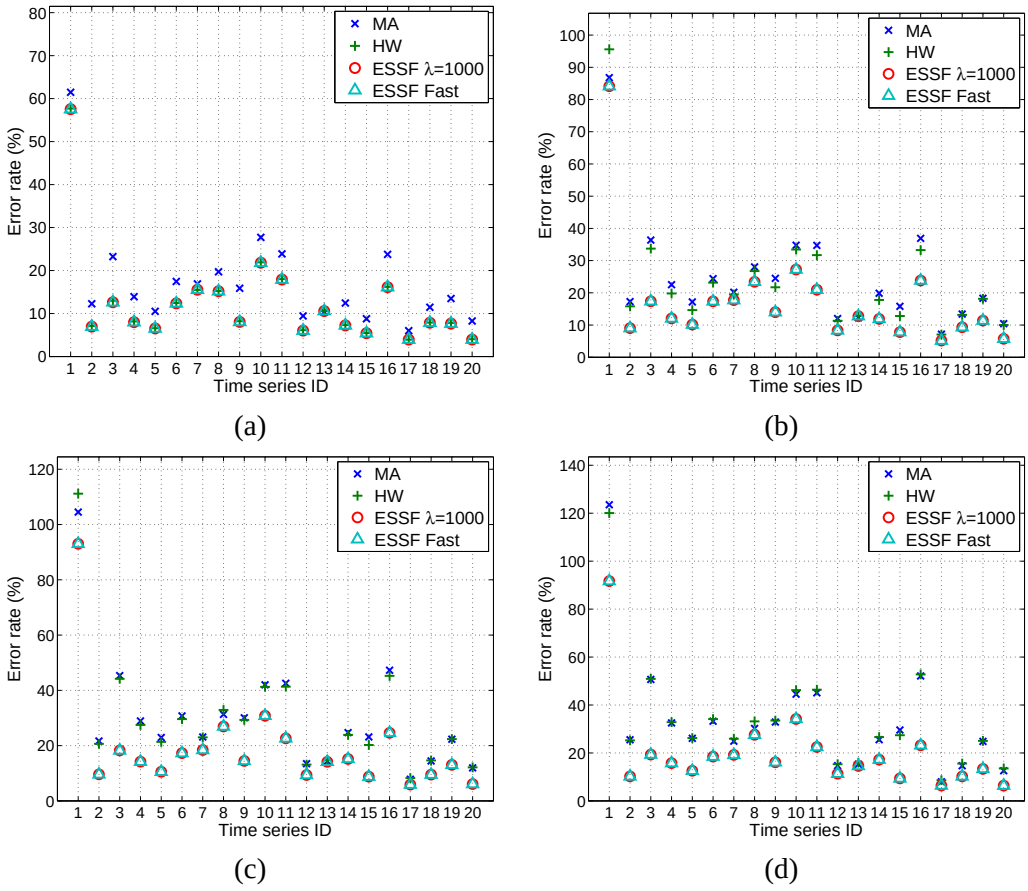


Figure 11: Compare the error rates by MA, HW, ESSF with $\lambda = 1000$, and ESSF Fast for twenty `g-trends` series. (a)-(d): The unit ahead $h = 1, 30, 60, 90$.

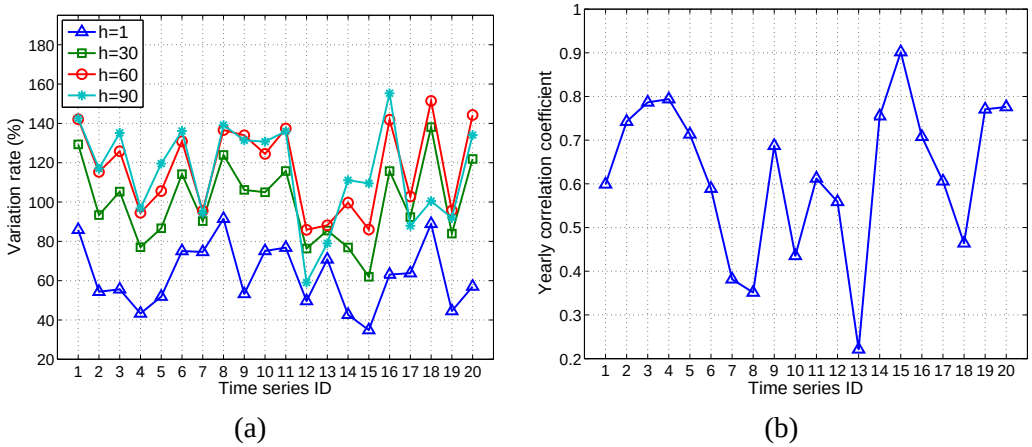


Figure 12: Predictability of twenty `g-trends` series. (a) The variation rates at $h = 1, 30, 60, 90$; (b) The average serial correlation coefficient between adjacent years.

ID	Query phrase	ID	Query phrase
1	American idol	11	human population
2	Anthropology	12	information technology
3	Aristotle	13	martial art
4	Art history	14	Monet
5	Beethoven	15	National park
6	Confucius	16	NBA
7	Cosmology	17	photography
8	cure cancer	18	public health
9	democracy	19	Shakespeare
10	financial crisis	20	Yoga

Table 2: The query phrases for twenty g-trends series and their IDs.

$Var(\cdot)$ denotes the serial variance. This rate is the ratio between the standard deviation of the change in page view h time units apart and that of the original series. A low variation rate indicates the series is less volatile and hence likely to be easier to predict. For example, Art history (ID 4) and National park (ID 15) have the lowest variation rates at $h = 1$, and they both yield relatively low prediction error rates, as shown by Figure 11. We also compute the variation rates for the page views of Web sites Auton, Wang, and citeseer at $h = 1$. They are respectively 100.0%, 88.1%, and 48.2%. This shows that the volatility of page views at these Web sites is in a similar range as that of the g-trends series.

In addition to the variation rate, the yearly correlation of the time series also indicates the potential for accurate prediction. For each of the twenty g-trends series, we compute the average of the correlation coefficients between segments of the series in adjacent years (i.e., 2004/05, 05/06, 06/07). Figure 12(b) shows the results. A series with high yearly correlation tends to benefit more in prediction from the yearly season extraction of ESSF. For instance, martial art (ID 13) has relatively low yearly correlation. The four prediction methods perform nearly the same for this series. In contrast, for NBA (ID 16) and democracy (ID 9), which have high yearly correlation, ESSF achieves substantially better prediction accuracy than HW and MA at all the values of h.

To compare the computational load of the prediction algorithms, we acquire the average user running time over the twenty g-trends series for one day ahead ($h = 1$) prediction at all the days in the last two years, 2006 and 2007. Again, we use Matlab codes on 2.4GHz Dell computer with Linux OS. The average time is respectively 0.11, 0.32, 0.59, and 3.77 seconds for MA, HW, ESSF Fast, and ESSF with $\lambda = 1000$.

6. Discussion and Conclusions

We have so far focused on extracting the trend and season parts of a time series using either HW or GLS, and have not considered predicting the noise part, as given in Eq. (1). We have argued that the variation in Web page view series is dominated by that of the trend and season. To quantitatively assess the potential gain from modeling and predicting the noise term in HW, we fit AR models to the noise. Specifically, we compute the level L_t and the season I_t by HW and let the noise $N_t = x_t - L_t - I_t$. We then fit AR models of order p to the noise series using the Yule-Walker estimation (Brockwell and Davis, 2002). We let p range from 1 to 10 and select an order p by the large-sample

motivated method described in Brockwell and Davis (1991, 2002). The fitted AR models are used to predict the noise, and the predicted noise is added to the forecasting value by HW. Suppose we want to predict x_{t+1} based on $\{x_1, x_2, ..., x_t\}$. The formula given by HW is $\hat{x}_{t+1} = L_t + I_{t+1-d}$. The predicted noise at $t+1$ given by the AR model is $\hat{N}_{t+1} = \hat{\phi}_1 N_t + \hat{\phi}_2 N_{t-1} + \cdots + \hat{\phi}_p N_{t-p+1}$, where $\hat{\phi}_j$, $j = 1, ..., p$, are estimated parameters in the AR model. We then adjust the prediction of HW by $\hat{x}_{t+1} = L_t + I_{t+1-d} + \hat{N}_{t+1}$.

In our experiments, the order of the AR model chosen for each of the three series `Auton`, `Wang`, and `citeseer` is 5, 6, 9 respectively. The error rates R_e^{adj} obtained for `Auton`, `Wang`, and `citeseer` are 17.16%, 20.30%, and 8.45%. As listed in Table 1, the error rates obtained by HW are 16.76%, 20.77%, and 8.80%. We see that the error rates for `Wang` and `citeseer` are improved via noise prediction, but that for `Auton` is degraded. For every series, the difference is insignificant. This shows that the gain from predicting the noise series is minor if positive at all. It is out of the scope of this paper to investigate more sophisticated models for the noise series. We consider it an interesting direction for future work.

To conclude, we have examined multiple approaches to Web page view forecasting. For short-term prediction, the HW procedure and the GLS state space model are investigated. It is shown that seasonal effect is important for page view forecasting. We developed a method to identify impulses and to reduce the decrease in prediction accuracy caused by them. The HW procedure, although computationally simple, performs closely to the GLS approach for predicting a small number of time units ahead. For predicting moderately distant future, the GLS model with smoother level terms tends to perform better. We developed the ESSF algorithm to extract global trend and scalable long-range season with smoothness regularization. It is shown that for predicting the distant future, ESSF outperforms HW significantly.

Acknowledgments

We thank Michael Baysek, C. Lee Giles, and James Wang for providing the logs of the Auton Lab, citeseer, and the Wang Lab. We also thank Artem Boytsov and Eyal Molad for helping us access the Google trends series, Robbie Sedgewick for suggestions on writing, and the reviewers for many insightful and constructive comments.

Appendix A. Algorithms for the State Space Model

Several major issues can be studied under the state space model:

1. Filtering: obtain the conditional distribution of α_{t+1} given X_t for $t = 1, ..., n$ where $X_t = \{x_1, ..., x_t\}$. If we consider α_t as the "true" signal, filtering is to discover the signal on the fly.

2. State smoothing: estimate α_t, $t = 1, ..., n$, given the entire series $\{x_1, ..., x_n\}$. This is to discover the signal in a batch mode.

3. Disturbance smoothing: estimate the disturbances $\hat{\varepsilon}_t = E(\varepsilon_t | y_1, ..., y_n)$, $\hat{\eta}_t = E(\eta_t | y_1, ..., y_n)$. The estimation can be used to estimate the covariance matrices of the disturbances.

4. Forecasting: given $\{x_1, ..., x_n\}$, forecast x_{n+j} for $j = 1, ..., J$.

5. Perform the Maximum Likelihood (ML) estimation for the parameters based on $\{x_1, ..., x_n\}$.

The computation methods involved in the above problems are tightly related. Filtering is conducted by forward recursion, while state smoothing is achieved by combining the forward recursion with a backward recursion. Disturbance smoothing can be easily performed based on the results of filtering and smoothing. ML estimation in turn relies on the result of disturbance smoothing. Next, we present the algorithms to solve the above problems.

A.1 Filtering and Smoothing

Recall the SSM described by Eq. (6)

$$\begin{aligned} x_t &= \mathbf{Z}_t\alpha_t + \varepsilon_t\,, \qquad \varepsilon_t \sim N(0,\mathbf{H}_t), \\ \alpha_{t+1} &= \mathbf{T}_t\alpha_t + \mathbf{R}_t\eta_t\,, \quad \eta_t \sim N(0,\mathbf{Q}_t)\,, \quad t = 1,\ldots,n, \\ &\qquad\qquad \alpha_1 \sim N(a_1,\mathbf{P}_1). \end{aligned}$$

Suppose the goal is filtering, that is, to obtain the conditional distribution of α_{t+1} given X_t for $t = 1,\ldots,n$ where $X_t = \{x_1,\ldots,x_t\}$. Since the joint distribution is Gaussian, the conditional distribution is also Gaussian and hence is uniquely determined by the mean and covariance matrix. Moreover, note that x_{t+1} is conditionally independent of X_t given α_{t+1}. Let $a_t = E(\alpha_t \mid X_{t-1})$ and $\mathbf{P}_t = Var(\alpha_t \mid X_{t-1})$. Then $\alpha_t \mid X_{t-1} \sim N(a_t,\mathbf{P}_t)$. It can be shown that a_{t+1} and $\mathbf{P}_{t+1}$ can be computed recursively from a_t, $\mathbf{P}_t$.

Let the one-step forecast error of x_t given X_{t-1} be v_t and the variance of v_t be $\mathbf{F}_t$:

$$\begin{aligned} v_t &= x_t - E(x_t \mid X_{t-1}) = x_t - \mathbf{Z}_t a_t, \\ \mathbf{F}_t &= Var(v_t) = \mathbf{Z}_t\mathbf{P}_t\mathbf{Z}_t^t + \mathbf{H}_t. \end{aligned}$$

For clarity, also define

$$\begin{aligned} \mathbf{K}_t &= \mathbf{T}_t\mathbf{P}_t\mathbf{Z}_t^t\mathbf{F}_t^{-1}, \\ \mathbf{L}_t &= \mathbf{T}_t - \mathbf{K}_t\mathbf{Z}_t. \end{aligned}$$

Then a_t, $\mathbf{P}_t$, $t = 2,\ldots,n+1$ can be computed recursively by updating v_t, $\mathbf{F}_t$, $\mathbf{K}_t$, $\mathbf{L}_t$, a_{t+1}, $\mathbf{P}_{t+1}$ as follows. It is assumed that a_1 and $\mathbf{P}_1$ are part of the model specification, and hence are known or provided by initialization. Details on initialization are referred to Durbin and Koopman (2001). For $t = 1,2,\ldots,n$,

$$\begin{aligned} v_t &= x_t - \mathbf{Z}_t a_t, \\ \mathbf{F}_t &= \mathbf{Z}_t\mathbf{P}_t\mathbf{Z}_t^t + \mathbf{H}_t, \\ \mathbf{K}_t &= \mathbf{T}_t\mathbf{P}_t\mathbf{Z}_t^t\mathbf{F}_t^{-1}, \\ \mathbf{L}_t &= \mathbf{T}_t - \mathbf{K}_t\mathbf{Z}_t, \\ a_{t+1} &= \mathbf{T}_t a_t + \mathbf{K}_t v_t, \\ \mathbf{P}_{t+1} &= \mathbf{T}_t\mathbf{P}_t\mathbf{L}_t^t + \mathbf{R}_t\mathbf{Q}_t\mathbf{R}_t^t. \end{aligned} \tag{17}$$

The above recursion is called *Kalman filter*. The dimensions for the above matrices are:

$$\begin{array}{ll} v_t & p \times 1\ , \\ \mathbf{F}_t & p \times p\ , \\ \mathbf{K}_t & m \times p\ , \\ \mathbf{L}_t & m \times m\ , \\ a_t & m \times 1\ , \\ \mathbf{P}_t & m \times m\ . \end{array}$$

We are concerned with univariate forecasting here with $p = 1$.

We now consider the *smooth estimation* $\hat{\alpha}_t = E(\alpha_t \mid x_1, x_2, ..., x_{t-1}, x_t, ..., x_n)$. Note its difference from the forward estimation $a_t = E(\alpha_t \mid x_1, x_2, ..., x_{t-1})$. The smooth estimation takes into consideration the series after t. Let the variance of the smooth estimation be $\mathbf{V}_t = Var(\alpha_t \mid x_1, x_2, ..., x_{t-1}, x_t, ..., x_n)$.

We can compute $\hat{\alpha}_t$ and $\mathbf{V}_t$ by the *backwards recursion* specified below. At $t = n$, set $\gamma_n = [0]_{m\times 1}$ and $\mathbf{N}_n = [0]_{m\times m}$. For $t = n, n-1, ..., 1$,

$$\begin{aligned} \gamma_{t-1} &= \mathbf{Z}_t^t \mathbf{F}_t^{-1} v_t + \mathbf{L}_t^t \gamma_t\ , \\ \mathbf{N}_{t-1} &= \mathbf{Z}_t^t \mathbf{F}_t^{-1} \mathbf{Z}_t + \mathbf{L}_t^t \mathbf{N}_t \mathbf{L}_t, \\ \hat{\alpha}_t &= a_t + \mathbf{P}_t \gamma_{t-1}, \\ \mathbf{V}_t &= \mathbf{P}_t - \mathbf{P}_t \mathbf{N}_{t-1} \mathbf{P}_t. \end{aligned} \tag{18}$$

Note that $\mathbf{Z}_t$, $\mathbf{F}_t$, $\mathbf{L}_t$, and $\mathbf{P}_t$ are already acquired by the Kalman filter (17). Eq. (17) and (18) are referred to as *Kalman filter and smoother*. The Kalman filter only involves forward recursion, while the smoother involves both forward and backward recursions.

A.2 Disturbance Smoothing

Let the smoothed disturbances be $\hat{\varepsilon}_t = E(\varepsilon_t | x_1, x_2, ..., x_n)$, $\hat{\eta}_t = E(\eta_t | x_1, x_2, ..., x_n)$. Suppose $\mathbf{F}_t$, $\mathbf{K}_t$, $\mathbf{L}_t$, $t = 1, ..., n$, have been obtained by the Kalman filter, and γ_t, $\mathbf{N}_t$ have been obtained by the Kalman smoother. Then we have

$$\begin{aligned} \hat{\varepsilon}_t &= \mathbf{H}_t(\mathbf{F}_t^{-1} v_t - \mathbf{K}_t^t \gamma_t), \\ Var(\varepsilon_t | x_1, x_2, ..., x_n) &= \mathbf{H}_t - \mathbf{H}_t(\mathbf{F}_t^{-1} + \mathbf{K}_t^t \mathbf{N}_t \mathbf{K}_t)\mathbf{H}_t, \\ \hat{\eta}_t &= \mathbf{Q}_t \mathbf{R}_t^t \gamma_t, \\ Var(\eta_t | x_1, x_2, ..., x_n) &= \mathbf{Q}_t - \mathbf{Q}_t \mathbf{R}_t^t \mathbf{N}_t \mathbf{R}_t \mathbf{Q}_t. \end{aligned}$$

A.3 Forecasting

Now suppose we want to forecast x_{n+j}, $j = 1, ..., J$, given $\{x_1, ..., x_n\}$. Let

$$\begin{aligned} \overline{x}_{n+j} &= E(x_{n+j} \mid x_1, x_2, ..., x_n), \\ \overline{\mathbf{F}}_{n+j} &= Var(x_{n+j} \mid x_1, x_2, ..., x_n)\ . \end{aligned}$$

First, we compute $\overline{a}_{n+j}$ and $\overline{P}_{n+j}$, $j = 1, ..., J$, by forward recursion similar to the Kalman filter in Eq. (17). The slight difference is that when $j = 1, ..., J-1$, set $v_{n+j} = 0$ and $\mathbf{K}_{n+j} = 0$. Specifically, set $\overline{a}_{n+1} = a_{n+1}$, $\overline{P}_{n+1} = \mathbf{P}_{n+1}$. The recursion for $\overline{a}_{n+j+1}$ and $\overline{P}_{n+j+1}$ for $j = 1, ..., J-1$ is:

$$\begin{aligned} \overline{a}_{n+j+1} &= \mathbf{T}_{n+j} \overline{a}_{n+j}, \\ \overline{\mathbf{P}}_{n+j+1} &= \mathbf{T}_{n+j} \overline{\mathbf{P}}_{n+j} \mathbf{T}_{n+j}^t + \mathbf{R}_{n+j} \mathbf{Q}_{n+j} \mathbf{R}_{n+j}^t\ . \end{aligned}$$

Then we forecast

$$\begin{aligned}\overline{x}_{n+j} &= \mathbf{Z}_{n+j}\overline{a}_{n+j},\\ \overline{\mathbf{F}}_{n+j} &= \mathbf{Z}_{n+j}\overline{\mathbf{P}}_{n+j}\mathbf{Z}_{n+j}^{t}+\mathbf{H}_{n+j}\,.\end{aligned}$$

A.4 Maximum Likelihood Estimation

The parameters to be estimated in the SSM are $\mathbf{H}_t$ and $\mathbf{Q}_t$, $t=1,2,...,n$. The EM algorithm is used to obtain the ML estimation. The missing data in EM in this case are the unobservable states α_t, $t=1,...,n$. Denote the parameters to be estimated collectively by ψ and the parameters obtained from the previous iteration by $\tilde{\psi}$. Let $\alpha=\{\alpha_1,\alpha_2,...,\alpha_n\}$ and $X_n=\{x_1,x_2,...,x_n\}$. The update of the EM algorithm comprises two steps:

1. Compute the expectation
$$E_{\tilde{\psi},X_n}[\log p(\alpha,X_n|\psi)]\,.$$

2. Maximize over ψ the above expectation.

It can be shown that

$$\begin{aligned}E_{\tilde{\psi},X_n}[\log p(\alpha,X_n|\psi)] &= \text{constant}-\frac{1}{2}\sum_{t=1}^{n}[\log|\mathbf{H}_t|+\log|\mathbf{Q}_{t-1}|+\\ &\quad tr[(\hat{\varepsilon}_t\hat{\varepsilon}_t^t+Var(\varepsilon_t|X_n))\mathbf{H}_t^{-1}]+\\ &\quad tr[(\hat{\eta}_{t-1}\hat{\eta}_{t-1}^t+Var(\eta_{t-1}|X_n))\mathbf{Q}_{t-1}^{-1}]\mid\psi]\end{aligned}$$

where $\hat{\varepsilon}_t$, $\hat{\eta}_{t-1}$, $Var(\varepsilon_t|X_n)$, and $Var(\eta_{t-1}|X_n)$ are computed by disturbance smoothing under parameter $\tilde{\psi}$. In the special case, when $\mathbf{H}_t=\mathbf{H}$, $\mathbf{Q}_t=\mathbf{Q}$, the maximization can be solved analytically:

$$\begin{aligned}\mathbf{H} &= \frac{\sum_{t=1}^{n}[\hat{\varepsilon}_t\hat{\varepsilon}_t^t+Var(\varepsilon_t|X_n)]}{n},\\ \mathbf{Q} &= \frac{\sum_{t=2}^{n}[\hat{\eta}_{t-1}\hat{\eta}_{t-1}^t+Var(\eta_{t-1}|X_n)]}{n-1}.\end{aligned}$$

The formula can be further simplified if $\mathbf{H}$ and $\mathbf{Q}$ are assumed diagonal. Suppose

$$\begin{aligned}\mathbf{H} &= diag(\sigma_{\varepsilon,1}^2,\sigma_{\varepsilon,2}^2,...,\sigma_{\varepsilon,p}^2),\\ \mathbf{Q} &= diag(\sigma_{\eta,1}^2,\sigma_{\eta,2}^2,...,\sigma_{\eta,r}^2)\,.\end{aligned}$$

Then

$$\begin{aligned}\sigma_{\varepsilon,j}^2 &= \frac{\sum_{t=1}^{n}[\hat{\varepsilon}_{t,j}^2+Var(\varepsilon_{t,j}|X_n)]}{n},\quad j=1,...,p,\\ \sigma_{\eta,j}^2 &= \frac{\sum_{t=2}^{n}[\hat{\eta}_{t-1,j}^2+Var(\eta_{t-1,j}|X_n)]}{n-1},\quad j=1,...,r\,.\end{aligned}$$

Appendix B. The ESSF Algorithm and Its Fast Version

To solve $\min_{\bar{\mathbf{y}},\mathbf{c}} G(\bar{\mathbf{y}},\mathbf{c})$ in Eq. (12), we iteratively optimize over $\bar{\mathbf{y}}$ and $\mathbf{c}$. Given $\mathbf{c}$, $\bar{\mathbf{y}}$ is solved by

$$\mathbf{A}_y \bar{\mathbf{y}} = \mathbf{b}_y$$

where $\mathbf{A}_y$ is a $D \times D$ matrix with non-zero entries:

$$\begin{aligned}
\mathbf{A}_y(j,j) &= \sum_k \alpha_{k,j}^2 + 6\lambda, \quad j = 1,2,...,D, \\
\mathbf{A}_y(j,j-1) &= \mathbf{A}_y(j,j+1) = -4\lambda, \quad j = 1,2,...,D, \\
\mathbf{A}_y(j,j-2) &= \mathbf{A}_y(j,j+2) = \lambda, \quad j = 1,2,...,D,
\end{aligned}$$

and the column vector $\mathbf{b}_y = (\sum_k \alpha_{k,j} z_{k,j})_j$. Recall that $\alpha_{k,j}$ is computed from $\mathbf{c}$ by Eq. (11).

Given $\mathbf{y}$, $\mathbf{c}$ is solved by

$$\mathbf{A}_c \mathbf{c} = \mathbf{b}_c$$

where $\mathbf{A}_c$ is a $K \times K$ matrix. Define $\mathbf{w}_1 = (0, \frac{1}{D}, \frac{2}{D}, ..., \frac{D-1}{D})^t$ and $\mathbf{w}_2 = (1, \frac{D-1}{D}, \frac{D-2}{D}, ..., \frac{1}{D})^t$. Let diagonal matrices $\mathbf{W}_1 = diag(\mathbf{w}_1)$, $\mathbf{W}_2 = diag(\mathbf{w}_2)$. Also define $\mathbf{z}_k = (z_{k,1}, z_{k,2}, ..., z_{k,D})^t$. The non-zero entries of $\mathbf{A}_c$ are:

$$\begin{aligned}
\mathbf{A}_c(k,k) &= (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{W}_1 \bar{\mathbf{y}} + (\mathbf{W}_2\bar{\mathbf{y}})^t \mathbf{W}_2 \bar{\mathbf{y}}, \quad k = 1,2,...,K-1, \\
\mathbf{A}_c(K,K) &= (\mathbf{W}_2\bar{\mathbf{y}})^t \mathbf{W}_2 \bar{\mathbf{y}}, \\
\mathbf{A}_c(k,k-1) &= (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{W}_2 \bar{\mathbf{y}}, \quad k = 2,3,...,K, \\
\mathbf{A}_c(k,k+1) &= (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{W}_2 \bar{\mathbf{y}}, \quad k = 1,2,...,K-1,
\end{aligned}$$

and the column vector $\mathbf{b}_c$ is given by:

$$\begin{aligned}
\mathbf{b}_c(1) &= (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{z}_1 + (\mathbf{W}_2\bar{\mathbf{y}})^t \mathbf{z}_2 - (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{W}_2 \bar{\mathbf{y}}, \\
\mathbf{b}_c(k) &= (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{z}_k + (\mathbf{W}_2\bar{\mathbf{y}})^t \mathbf{z}_{k+1}, \quad k = 2,3,...,K-1, \\
\mathbf{b}_c(K) &= (\mathbf{W}_1\bar{\mathbf{y}})^t \mathbf{z}_1.
\end{aligned}$$

In summary, the ESSF algorithm iterates the following two steps with initialization $\mathbf{c}^{(0)} = \mathbf{1}$. At iteration $p \geq 1$:

1. Given $\mathbf{c}^{(p-1)}$, compute $\mathbf{A}_y$ and $\mathbf{b}_y$. Let $\bar{\mathbf{y}}^{(p)} = \mathbf{A}_y^{-1}\mathbf{b}_y$.
2. Given $\bar{\mathbf{y}}^{(p)}$, compute $\mathbf{A}_c$ and $\mathbf{b}_c$. Let $\mathbf{c}^{(p)} = \mathbf{A}_c^{-1}\mathbf{b}_c$.

For the fast version of ESSF, we need to solve $\min_{\bar{\mathbf{y}},\mathbf{c}} \tilde{G}(\bar{\mathbf{y}},\mathbf{c})$ in Eq. (13). We start with $\mathbf{c}^{(0)} = \mathbf{1}$. Without loss of generality, we fix $c_1 = 1$. At iteration $p \geq 1$:

1. Given $\mathbf{c}^{(p-1)}$, compute
$$\bar{y}_j^{(p)} = \frac{\sum_k c_k^{(p-1)} z_{k,j}}{\| \mathbf{c}^{(p-1)} \|^2}, \quad j = 1,...,D.$$
2. Given $\bar{\mathbf{y}}^{(p)}$, compute
$$c_k^{(p)} = \frac{\sum_j z_{k,j} \bar{y}_j^{(p)}}{\| \bar{\mathbf{y}}^{(p)} \|^2}, \quad k = 1,...,K.$$

References

B. D. O. Anderson and J. B. Moore. *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

A. Aussem and F. Murtagh. Web traffic demand forecasting using wavelet-based multiscale decomposition. *International Journal of Intelligent Systems*, 16(2):215-236, 2001.

S. Basu, A. Mukherjee, and S. Klivansky. Time series models for Internet traffic. *INFOCOM '96. Fifteenth Annual Joint Conference of the IEEE Computer Societies, Networking the Next Generation*, 611-620, 1996.

G. E. P. Box and G. M. Jenkins. *Time-Series Analysis, Forecasting and Control*, San Francisco: Holden-Day, 1970.

P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods*, 2nd Edition, Springer-Verlag, New York, 1991.

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*, 2nd Edition, Springer Science+Business Media, Inc., New York, 2002.

C. Chatfield. *The Analysis of Time Series*, Chapman & Hall/CRC, New York, 2004.

J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods*, Oxford University Press Inc., New York, 2001.

M. Grossglauser and J.-C. Bolot. On the relevance of long-range dependence in network traffic. *IEEE/ACM Transactions Networking*, 7(5):629-640, 1999.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering, Series D*, 82:35-45, 1960.

A. Khotanzad and N. Sadek. Multi-scale high-speed network traffic prediction using combination of neural networks. *Proc. Int. Joint Conf. Neural Networks*, 2:1071-1075, July 2003.

A. M. Odlyzko. Internet traffic growth: sources and implications. *Optical Transmission Systems and Equipment for WDM Networking II*, B. B. Dingel, W. Weiershausen, A. K. Dutta, and K.-I. Sato, eds.,*Proc. SPIE*, 5247:1-15, 2003.

K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot. Long-term forecasting of Internet backbone traffic. *IEEE Trans. Neural Networks*, 16(5):1110-1124, 2005.

K. Park and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation*, John Wiley & Sons, Inc., 2000.

A. P. Sage and J. L. Melsa. *Estimation Theory with Applications to Communication and Control*, McGraw Hill, New York, 1971.

A. Sang and S. Li. A predictability analysis of network traffic. *Computer Networks*, 39(4):329-345, 2002.

W. W. S. Wei. *Time Series Analysis, Univariate and Multivariate Methods*, 2nd Edition, Pearson Education, Inc., 2006.

C. You and K. Chandra. Time series models for Internet data traffic. *Proc. 24th Annual IEEE Int. Conf. Local Computer Networks (LCN'99)*, 164, 1999.

Journal of Machine Learning Research 9 (2008) 2251-2286 Submitted 12/07; Revised 6/08; Published 10/08

Finding Optimal Bayesian Network Given a Super-Structure

Eric Perrier PERRIER@IMS.U-TOKYO.AC.JP
Seiya Imoto IMOTO@IMS.U-TOKYO.AC.JP
Satoru Miyano MIYANO@IMS.U-TOKYO.AC.JP
Human Genome Center, Institute of Medical Science
University of Tokyo
4-6-1 Shirokanedai, Minato-ku, Tokyo 108-8639, Japan

Editor: Max Chickering

Abstract

Classical approaches used to learn Bayesian network structure from data have disadvantages in terms of complexity and lower accuracy of their results. However, a recent empirical study has shown that a hybrid algorithm improves sensitively accuracy and speed: it learns a skeleton with an independency test (IT) approach and constrains on the directed acyclic graphs (DAG) considered during the search-and-score phase. Subsequently, we theorize the structural constraint by introducing the concept of super-structure S, which is an undirected graph that restricts the search to networks whose skeleton is a subgraph of S. We develop a super-structure constrained optimal search (COS): its time complexity is upper bounded by $O(\gamma_m{}^n)$, where $\gamma_m < 2$ depends on the maximal degree m of S. Empirically, complexity depends on the average degree $\tilde{m}$ and sparse structures allow larger graphs to be calculated. Our algorithm is faster than an optimal search by several orders and even finds more accurate results when given a sound super-structure. Practically, S can be approximated by IT approaches; significance level of the tests controls its sparseness, enabling to control the trade-off between speed and accuracy. For incomplete super-structures, a greedily post-processed version (COS+) still enables to significantly outperform other heuristic searches.

Keywords: Bayesian networks, structure learning, optimal search, super-structure, connected subset

1. Introduction

It is impossible to understand large raw sets of data obtained from a huge number of correlated variables. Therefore, in order to simplify the comprehension of the system, various graphical models have been developed to summarize interactions between such variables in a synoptic graph. Among the existing models, Bayesian networks have been widely employed for decades in various domains including artificial intelligence (Glymour, 2001), medicine (Cowell et al., 1999), bioinformatics (Friedman et al., 2000), and even economy (Segal et al., 2005) and sociology (Heckerman, 1996). Bayesian networks compactly represent a joint probability distribution P over the set of variables, using DAG to encode conditional independencies between them (Pearl, 1988). The popularity of this model is primarily due to its high expressive power, enabling the simultaneous investigation of complex relationships between many variables of a heterogeneous nature (discrete or continuous). Further, for Bayesian network model inference from data is comparatively simpler; incomplete or noisy data are also usable and prior knowledge can be incorporated. When the DAG or structure of

the model is known, the parameters of the conditional probability distributions can be easily fit to the data; thus, the bottleneck of modeling an unknown system is to infer its structure.

Over the previous decades, various research directions have been explored through a numerous literature to deal with structure learning, which let us propose the following observations. Maximizing a score function over the space of DAGs is a promising approach towards learning structure from data. A search strategy called optimal search (OS) have been developed to find the graphs having the highest score (or global optima) in exponential time. However, since it is feasible only for small networks (containing up to thirty nodes), in practice heuristic searches are used. The resulting graphs are local optima and their accuracy strongly depends on the heuristic search strategy. In general, given no prior knowledge, the best strategy is still a basic greedy hill climbing search (HC). In addition, Tsamardinos et al. (2006) proposed to constrain the search space by learning a skeleton using an IT-based technique before proceeding to a restricted search. By combining this method with a HC search, they developed a hybrid algorithm called max-min hill-climbing (MMHC) that is faster and usually more accurate.

In the present study we are interested in OS since the optimal graphs will converge to the true model in the sample limit. We aim to improve the speed of OS in order to apply it to larger networks; for this, a structural constraint could be of a valuable help. In order to keep the asymptotic correctness of OS, the constraint has to authorize at least the edges of the true network, but it can contain also extra edges. Following this minimal condition that should respect a constraint on the skeletons to be sound, we formalize a flexible structural constraint over DAGs by defining the concept of a super-structure. This is an undirected graph that is assumed to contain the skeleton of the true graph (i.e., the true skeleton). In other word, the search space is the set of DAGs that have a subgraph of the given super-structure as a skeleton. A sound super-structure (that effectively contains the true skeleton) could be provided by prior knowledge or learned from data much more easily (with a higher probability) than the true skeleton itself. Subsequently, we consider the problem of maximizing a score function given a super-structure, and we derive a constrained optimal search, COS, that finds a global optimum over the restricted search space. Not surprisingly, our algorithm is faster than OS since the search space is smaller; more precisely, its computational complexity is proportional to the number of connected subsets of the super-structure. An upper bound is derived theoretically and average complexity is experimentally showed to depend on the average degree of the super-structure. Concretely, for sparse structures our algorithm can be applied to larger networks than OS (with an average degree around 2.1, graphs having 1.6 times more nodes could be considered). Moreover, for a sound super-structure, learned graphs are more accurate than unconstrained optima: this is because, some incorrect edges are forbidden, even if their addition to the graph improves the score.

Since the sparseness directly affects the speed, and therefore the feasibility of our search, it remains to propose efficient methods to learn a sound and sparse super-structures without prior knowledge. This is out of the scope of this present paper where we focus on the enunciation of our constraint, its application to optimal search and optimizations of its implementation. Nevertheless, in order to demonstrate our algorithm in practice, we propose a first basic strategy to approximate a super-structure from data. The idea is to use "relaxed" independency testing to obtain an undirected graph that may contain the true skeleton with a high probability, while yet being sparse. In that case, we can consider the significance level of the independency tests, α, as a tool to choose between accuracy (high values return dense but probably sound structures) and speed (low values give sparse but incomplete structures). We tested our proposition on MMPC, the IT-based strategy

used by Tsamardinos et al. (2006) in MMHC; our choice was motivated by the good results of their algorithm that we also include in our comparative study. MMPC appears to be a good method to learn robust and relatively sparse skeletons; unfortunately, soundness is achieved only for high significance levels, $\alpha > 0.9$, implying a long calculation and a denser structure. Practically, when the constraint is learned with $\alpha = 0.05$, in terms of accuracy, COS is worse than OS since the super-structure is usually incomplete; still, COS outperforms most of the time greedy searches, although it finds graphs of lower scores. Resulting graphs can be quickly improved by applying to them a post-processing unconstrained hill-climbing (COS+). During that final phase, scores are strictly improved, and usually accuracy also. Interestingly, even for really low significance levels ($\alpha \approx 10^{-5}$), COS+ returns graphs more accurate and of a higher score than both MMHC and HC. COS+ can be seen as a bridge between HC (when α tends to 0) and OS (when α tends to 1) and can be applied up to a hundred nodes by selecting a low enough significance level.

This paper is organized as follows. In Section 2, we discuss the existing literature on structure learning. We clarify our notation in Section 3.1 and reintroduce OS in Section 3.2. Then, in Section 4, the core of this paper, we define super-structures and present our algorithm, proofs of its complexity and practical information for implementation. Section 5 details our experimental procedures and presents the results. Section 5.1.4 briefly recalls MMPC, the method we used during experiments to learn the super-structures from data. Finally, in Section 6, we conclude and outline our future works.

2. Related Works

The algorithms for learning the Bayesian network structure that have been proposed until now can be regrouped into two different approaches, which are described below.

2.1 IT Approach

This approach includes IC algorithm (inductive causation) (Pearl, 1988), PC algorithm (after its authors, Peter and Clark) (Spirtes et al., 2000), GS algorithm (grow and shrink) (Margaritis and Thrun, 2000), and TPDA algorithm (three-phase dependency analysis) (Cheng et al., 2002). All of them build the structure to be consistent with the conditional independencies among the variables that are evaluated with a statistical test (G-square, partial correlation). Usually, algorithms start by learning the skeleton of the graph (by propagating constraints on the neighborhood of each variable) and then edges are oriented to cope with dependencies revealed from data. Finally, one network is retained from the equivalent class consistent with the series of tests. Under the faithful condition of *P*, such strategies have been proven to build a graph converging to the true network as the size of the data approaches infinity. Moreover, their complexity is polynomial, assuming that the maximal degree of the network, that is, the maximal size of nodes neighborhood, is bounded (Kalisch and Bühlmann, 2007). However, in practice, the results are mixed because of the tests sensitivity to noise: since these algorithms base their decisions on a single or few tests, they are prone to accumulate errors (Margaritis and Thrun, 2000). Worse, they can obtain a set of conditional independencies that is contradictory, or that cannot be faithfully encoded by a DAG, leading to a failure of the algorithm. Moreover, except for sparse graphs, their execution time is generally longer than that of algorithms from the scoring criteria-based approach (Tsamardinos et al., 2006).

2.2 Scoring Criteria-Based Approach

Search-and-score methods are favored in practice and considered as a more promising research direction. This second family of algorithms uses a scoring criterion, such as the posterior probability of the network given the data, in order to evaluate how well a given DAG fits empirical results, and returns the one that maximized the scoring function during the search. Since the search space is known to be of a super exponential size on the number of nodes n, that is, $O(n!2^{\binom{n}{2}})$ (Robinson, 1973), an exhaustive search is practically infeasible, implying that various greedy strategies have been proposed to browse DAG space, sometimes requiring some prior knowledge.

Among them, the state-of-the-art greedy hill climbing (HC) strategy, although it is simple and will find only a locally optimal network, remains one of the most employed method in practice, especially with larger networks. There exist various implementations using different empirical tricks to improve the score of the results, such as TABU list, restarting, simulated annealing, or searching with different orderings of the variables (Chickering et al., 1995; Bouckaert, 1995). However a traditional and basic algorithm will process in the following manner:

- Start the search from a given DAG, usually the empty one.
- Then, from a list of possible transformations containing at least addition, withdrawal or reversal of an edge, select and apply the transformation that improves the score most while also ensuring that graph remains acyclic.
- Finally repeat previous step until strict improvements to the score can no longer be found.

More details about our implementation of HC are given in Section 5.1.3. Such an algorithm can be used even for large systems, and if the number of variables is really high, it can be adapted by reducing the set of transformations considered, or by learning parents of each node successively. In any case, this algorithm finds a local optimum DAG but without any assertion about its accuracy (besides its score). Further, the result is probably far from a global optimal structure, especially when number of nodes increases. However, optimized forms of this algorithm obtained by using one or more tricks have been considered to be the best search strategies in practice until recently.

Other greedy strategies have also been developed in order to improve either the speed or accuracy of HC one: sparse candidate (SC, Friedman et al., 1999) that limits the maximal number of parents and estimate candidate parents for each node before the search, greedy equivalent search (GES, Chickering, 2002b) that searches into the space of equivalence classes (PDAGs), and optimal reinsertion (OR, Moore and Wong, 2003) that greedily applies an optimal reinsertion transformation repeatedly on the graph.

SC was one of the first to propose a reduction in the search space, thereby sensitively improving the score of resulting networks without increasing the complexity too much if candidate parents are correctly selected. However, it has the disadvantage of a lack of flexibility, since imposing a constant number of candidate parents to every node could be excessive or restrictive. Furthermore, the methods and measures proposed to select the candidates, despite their intuitive interest, have not been proved to include at least the true or optimal parents for each node.

GES has the benefit that it exploits a theoretically justified direction. Main scoring functions have been proved to be score equivalent (Chickering, 1995), that is, two equivalent DAGs (representing the same set of independencies among variables) have the same score. Thus they define

equivalent classes over networks that can be uniquely represented by CPDAGs. Therefore, searching into the space of equivalent classes reduces the number of cases that have to be considered, since one CPDAG represents several DAGs. Further, by using usual sets of transformations adapted to CPDAGs, the space browsed during a greedy search becomes more connected, increasing the chances of finding a better local maximum. Unfortunately, the space of equivalent classes seems to be of the same size order than that of DAGs, and an evaluation of the transformations for CPDAGs is more time consuming. Thus, GES is several times slower than HC, and it returns similar results. Interestingly, following the comparative study of Tsamardinos et al. (2006), if structural errors rather than scores are considered as a measure of the quality of the results, GES is better than a basic HC.

In the case of OR, the algorithm had the advantage to consider a new transformation that globally affects the graph structure at each step: this somehow enables the search to escape readily from local optima. Moreover, the authors developed efficient data-structures to rationalize score evaluations and retrieve easily evaluation of their operators. Thus, it is one of the best greedy methods proposed; however, with increasing data, the algorithm will collapse due to memory shortage.

Another proposed direction was using the K2 algorithm (Cooper and Herskovits, 1992), which constraints the causal ordering of variables. Such ordering can be seen to be a topological ordering of the true graph, provided that such a graph is acyclic. Based on this, the authors proposed a strategy to find an optimal graph by selecting the best parent set of a node among the subsets of nodes preceding it. The resulting graph can be the global optimal DAG if it accepts the same topological ordering. Therefore, given an optimal ordering, K2 can be seen as an optimal algorithm with a time and space complexity of $O(2^n)$. Moreover, for some scoring functions, branch-pruning can be used while looking for the best parent set of a node (Suzuki, 1998), thereby improving the complexity. However, in practice, a greedy search that considers adding and withdrawing a parent is applied to select a locally optimal parent set. In addition, the results are strongly depending on the quality of the ordering. Some investigations have been made to select better orderings (Teyssier and Koller, 2005) with promising results.

2.3 Recent Progress

One can wonder about the feasibility of finding a globally optimal graph without having to explicitly check every possible graph, since nothing can be asserted with respect to the structural accuracy of the local maxima found by previous algorithms. In a general case, learning Bayesian network from data is an NP-hard problem (Chickering, 1996), and thus for large networks, only such greedy algorithms are used. However, recently, algorithms for global optimization or exact Bayesian inference have been proposed (Ott et al., 2004; Koivisto and Sood, 2004; Singh and Moore, 2005; Silander and Myllymäki, 2006) and can be applied up to a few tens of nodes. Since they all principally share the same strategy that we will introduce in detail subsequently, we will refer to it as optimal search (OS). Even if such a method cannot be of a great use in practice, it could validate empirically the search-and-score approach by letting us study how a global maximum converges to the true graph when the data size increases; Also, it could be a meaningful gold standard to judge the performances of greedy algorithms.

Finally, a recent noteworthy step was performed with the min-max hill climbing algorithm (MMHC, Tsamardinos et al., 2006), since it was empirically proved to be the fastest and the best method in terms of structural error based on the structural hamming distance. This algorithm can be considered as a hybrid of the two approaches. It first learns an approximation of the skeleton of

the true graph by using an IT strategy. It is based on a subroutine called min-max-parents-children (MMPC) that reconstructs the neighborhood of each node; G-square tests are used to evaluate conditional independencies. The algorithm subsequently proceeds to a HC search to build a DAG limiting edge additions to the one present in the retrieved skeleton. As a result, it follows a similar technique than that of SC, except that the number of candidate parents is tuned adaptively for each node, and that the chosen candidates are sound in the sample limit. It is worth to notice that the skeleton learned in the first phase can differ from the one of the final DAG, since all edges will not be for sure added during the greedy search. However, it will be certainly a cover of the resulting graph skeleton.

3. Definitions and Preliminaries

In this section, after explaining our notations and recalling some important definitions and results, we discuss structure constraining and define the concept of a super-structure. Section 3.3 is dedicated to OS.

3.1 Notation and Results for Bayesian Networks

In the rest of the paper, we will use upper-case letters to denote random variables (e.g., X_i, V_i) and lower-case letters for the state or value of the corresponding variables (e.g., x_i, v_i). Bold-face will be used for sets of variables (e.g., $\mathbf{Pa}_i$) or values (e.g., $\mathbf{pa}_i$). We will deal only with discrete probability distributions and complete data sets for simplicity, although a continuous distribution case could also be considered using our method.

Given a set $\mathbf{X}$ of n random variables, we would like to study their probability distribution P_0. To model this system, we will use Bayesian networks:

Definition 1. (Pearl, 1988; Spirtes et al., 2000; Neapolitan, 2003) *Let P be a discrete joint probability distribution of the random variables in some set* $\mathbf{V}$*, and* $G = (\mathbf{V}, \mathbf{E})$ *be a directed acyclic graph (DAG). We call* (G, P) *a* Bayesian network (BN) *if it satisfies the Markov condition, that is, each variable is independent of any subset of its non-descendant variables conditioned on its parents.*

We will denote the set of the parents of a variable V_i in a graph G by $\mathbf{Pa}_i$, and by using the Markov condition, we can prove that for any BN (G, P), the distribution P can be factored as follows:

$$P(\mathbf{V}) = P(V_1, \cdots, V_p) = \prod_{V_i \in \mathbf{V}} P(V_i | \mathbf{Pa}_i).$$

Therefore, to represent a BN, the graph G and the joint probability distribution have to be encoded; for the latter, every probability $P(V_i = v_i | \mathbf{Pa}_i = \mathbf{pa}_i)$ should be specified. G directly encodes some of the independencies of P and entails others (Neapolitan, 2003). More precisely, all independencies entailed in a graph G are summarized by its skeleton and by its v-structures (Pearl, 1988). Consequently, two DAGs having the same skeleton and v-structures entail the same set of independencies; they are said to be *equivalent* (Neapolitan, 2003). This equivalence relation defines *equivalent classes* over space of DAGs that are unambiguously represented by *completed partially directed acyclic graphs (CPDAG)* (Chickering, 2002b). Finally, if all and only the conditional independencies true in a distribution P are entailed by the Markov condition applied to a DAG G, we say that the Bayesian Network (G, P) is *faithful* (Spirtes et al., 2000).

In our case, we will assume that the probability distribution P_0 over the set of random variables $\mathbf{X}$ is faithful, that is, that there exists a graph G_0, such that (G_0, P_0) is a faithful Bayesian network. Although there are distributions P that do not admit a faithful BN (for example the case when parents are connected to a node via a parity or XOR structure), such cases are regarded as "rare" (Meek, 1995), which justifies our hypothesis.

To study $\mathbf{X}$, we are given a set of data $\mathbf{D}$ following the distribution P_0, and we try to learn a graph G, such that (G, P_0) is a faithful Bayesian network. The graph we are looking for is probably not unique because any member of its equivalent class will also be correct; however, the corresponding CPDAG is unique. Since there may be numerous graphs G to which P_0 is faithful, several definitions are possible for the problem of learning a BN. We choose as Neapolitan (2003):

Definition 2. *Let P_0 be a faithful distribution and* $\mathbf{D}$ *be a statistical sample following it. The problem of learning the structure of a Bayesian network given* $\mathbf{D}$ *is to* induce a graph G so that (G, P_0) is a faithful BN, *that is, G and G_0 are on the same equivalent class, and both are called the true structure of the system studied.*

In every IT-based or constraint-based algorithm, the following theorem is useful to identify the skeleton of G_0:

Theorem 1. (Spirtes et al., 2000) *In a faithful BN (G, P) on variables* $\mathbf{V}$*, there is an edge between the pair of nodes X and Y if and only if X depends on Y conditioning on every subset* $\mathbf{Z}$ *included in* $\mathbf{V} \setminus \{X, Y\}$.

Thus, from the data, we can estimate the skeleton of G_0 by performing conditional independency tests (Glymour and Cooper, 1999; Cheng et al., 2002). We will return to this point in Section 4.1 since higher significance levels for the test could be used to obtain a cover of the skeleton of the true graph.

3.2 General Optimal Search

Before presenting our algorithm, we should review the functioning of an OS. Among the few articles on optimal search (Ott et al., 2004; Koivisto and Sood, 2004; Singh and Moore, 2005; Silander and Myllymäki, 2006), Ott and Miyano (2003) are to our knowledge the first to have published an exact algorithm. In this section we present the algorithm of Ott et al. (2004) for summarizing the main idea of OS. While investigating the problem of exact model averaging, Koivisto and Sood (2004) independently proposed another algorithm that also learn optimal graphs proceeding on a similar way. As for Singh and Moore (2005), they presented a recursive implementation that is less efficient in terms of calculation; however, it has the advantage that potential branch-pruning rules can be applied. Finally, Silander and Myllymäki (2006) detailed a practically efficient implementation of the search: the main advantage of their algorithm is to calculate efficiently the scores by using contingency tables (still computational complexity remains the same). They empirically demonstrated that optimal graphs could be learned up to $n = 29$.

To understand how OS finds global optima in $O(n2^n)$ without having to explicitly check every DAG possible, we must first explain how a score function is defined. Various scoring criteria for graphs have been defined, including Bayesian Dirichlet (specifically BDe with uniform priors, BDeu) (Heckerman et al., 1995), Bayesian information criterion (BIC) (Schwartz, 1978), Akaike information criterion (AIC) (Akaike, 1974), minimum description length (MDL) (Rissanen, 1978),

and Bayesian network and nonparametric regression criterion (BNRC) (Imoto et al., 2002). They are usually costly to evaluate; however, due to the Markov condition, they can be evaluated locally:

$$Score(G,\mathbf{D}) = \sum_{i=1}^{n} score(X_i, \mathbf{Pa}_i, \mathbf{D}).$$

This property is essential to enable efficient calculation, particularly with large graphs, and is usually supposed while defining an algorithm. Another classical attribute is score equivalence, which means that two equivalent graphs will have the same score. It was proved to be the case for BDe, BIC, AIC, and MDL (Chickering, 1995). In our study, we will use BIC, thereby our score is local and equivalent, and our task will be to find a DAG over $\mathbf{X}$ that maximizes the score given the data $\mathbf{D}$. Exploiting score locality, Ott et al. (2004) defined for every node X_i and every candidate parent set $\mathbf{A} \subseteq \mathbf{X} \setminus \{X_i\}$:

- The best local score on X_i: $F_s(X_i,\mathbf{A}) = \max_{\mathbf{B}\subseteq\mathbf{A}} score(X_i,\mathbf{B},\mathbf{D})$;
- The best parent set for X_i: $F_p(X_i,\mathbf{A}) = \operatorname{argmax}_{\mathbf{B}\subseteq\mathbf{A}} score(X_i,\mathbf{B},\mathbf{D})$.

From now we omit writing $\mathbf{D}$ when referring to the score function. F_s can be calculated recursively on the size of $\mathbf{A}$ using the following formulas:

$$F_s(X_i,\emptyset) = score(X_i,\emptyset), \quad (1)$$
$$F_s(X_i,\mathbf{A}) = \max(score(X_i,\mathbf{A}), \max_{X_j\in\mathbf{A}}(F_s(X_i,\mathbf{A}\backslash\{X_j\})). \quad (2)$$

Calculation of F_p directly follows; we will sometimes use F as a shorthand to refer to these two functions. Noticing that we can dynamically evaluate F, one can think that it is thus directly possible to find the best DAG. However, it is also essential to verify that the graph obtained is acyclic and hence, that there exists a topological ordering over the variables.

Definition 3. *Let w be an ordering defined on $\mathbf{A} \subseteq \mathbf{X}$ and $H = (\mathbf{A},\mathbf{E})$ be a DAG. We say that H is* w-linear *if and only if $w(X_i) < w(X_j)$ for every directed edge $(X_i,X_j) \in \mathbf{E}$.*

By using F_p and given an ordering w on $\mathbf{A}$ we derive the best w-linear graph G^*_w as:

$$G^*_w = (\mathbf{A},\mathbf{E}^*_w), \text{ with } (X_j,X_i) \in \mathbf{E}^*_w \text{ if and only if } X_j \in F_p(X_i, \mathbf{Pred}_w(X_i)). \quad (3)$$

Here, G^*_w is directly obtained by selecting for each variable $X_i \in \mathbf{A}$ its best parents among the nodes preceding X_i in the ordering w referred as $\mathbf{Pred}_w(X_i) = \{X_j \text{ with } w(X_j) < w(X_i)\}$. Therefore, to achieve OS, we need to find an optimal w^*, that is, a topological ordering of an optimal DAG. With this end, we define for every subset $\mathbf{A} \subseteq \mathbf{X}$ not empty:

- The best score of graphs G on $\mathbf{A}$: $M_s(\mathbf{A}) = \max_G Score(G)$
- The last node of an optimal ordering on $\mathbf{A}$: $M_l(\mathbf{A})$

Another way to interpret $M_l(\mathbf{A})$ is as a sink of an optimal graph on $\mathbf{A}$, that is, a node that has no children. M_s and M_l are simply initialized by:

$$\begin{aligned} \forall X_i \in \mathbf{X}: \;\; & M_s(\{X_i\}) = score(X_i, \emptyset), \\ & M_l(\{X_i\}) = X_i. \end{aligned} \tag{4}$$

When $|\mathbf{A}| = k > 1$, we consider an optimal graph G^* on that subset and w^* one of its topological ordering. The parents of the last element X_{i^*} are for sure $F_p(X_{i^*}, \mathbf{B}_{i^*})$, where $\mathbf{B}_j = \mathbf{A} \setminus \{X_j\}$; thus its local score is $F_s(X_{i^*}, \mathbf{B}_{i^*})$. Moreover, the subgraph of G^* induced when removing X_{i^*} must be optimal for $\mathbf{B}_{i^*}$; thus, its score is $M_s(\mathbf{B}_{i^*})$. Therefore, we can derive a formula to define M_l recursively:

$$M_l(\mathbf{A}) = X_{i^*} = \underset{X_j \in \mathbf{A}}{\operatorname{argmax}}(F_s(X_j, \mathbf{B}_j) + M_s(\mathbf{B}_j)). \tag{5}$$

This also enables us to calculate M_s directly. We will use M to refer to both M_s and M_l. M can be computed dynamically and M_l enables us to build quickly an optimal ordering w^*; elements are find in reverse order:

$$\begin{aligned} & \mathbf{T} = \mathbf{X} \\ & \text{While } \mathbf{T} \neq \emptyset \\ & \qquad w^*(M_l(\mathbf{T})) = |\mathbf{T}| \\ & \qquad \mathbf{T} = \mathbf{T} \setminus M_l(\mathbf{T}) \end{aligned} \tag{6}$$

Therefore, the OS algorithm is summarized by:

Algorithm 1 (OS). (Ott et al., 2004)

(a) Initialize $\forall X_i \in \mathbf{X}$, $F_s(X_i, \emptyset)$ and $F_p(X_i, \emptyset)$ with (1)

(b) For each $X_i \in \mathbf{X}$ and each $\mathbf{A} \subseteq \mathbf{X} \setminus \{X_i\}$:
Calculate $F_s(X_i, \mathbf{A})$ and $F_p(X_i, \mathbf{A})$ using (2)

(c) Initialize $\forall X_i$, $M_s(\{X_i\})$ and $M_l(\{X_i\})$ using (4)

(d) For each $\mathbf{A} \subseteq \mathbf{X}$ with $|\mathbf{A}| > 1$:
Calculate $M_s(\mathbf{A})$ and $M_l(\mathbf{A})$ using (5)

(e) Build an optimal ordering w^* using (6)

(f) Return the best w^*-linear graph $G^*_{w^*}$ using (3)

Note that in steps (b) and (d) subsets $\mathbf{A}$ are implicitly considered by increasing size to enable formulae (2) and (5). With respect to computational complexity, in steps (a) and (b) F is calculated for $n2^{n-1}$ pairs of variable and parent candidate set. In each case, one score exactly is computed. Then, M is computed over the 2^n subsets of $\mathbf{X}$ (step (c) and (d)). w^* and G^*_w are both build in $O(n)$ time at step (e) and (f); thus, the algorithm has a total time complexity of $O(n2^n)$ and evaluates $n2^{n-1}$ scores. Here, time complexity refers to the number of times that the formulae (2) or (5) are computed; however, it should be pointed out that these formulae require at least $O(n)$ basic operations.

As proposed (Ott et al., 2004), OS can be speed up by constraining with a constant c the maximal size of parent sets. This limitation is easily justifiable, as graphs having many parents for a node are usually strongly penalized by score functions. In that case, the computational complexity remains the same; only formulas (2) is constrained, and $score(X_i, \mathbf{A})$ is not calculated when $|\mathbf{A}| > c$. Consequently, the total number of score evaluated is reduced to $O(n^{c+1})$, which is a decisive improvement since computing a score is costly.

The space complexity of Algorithm 1 can be slightly reduced by recycling memory as mentioned (Ott et al., 2004). In fact, when calculating functions F and M for subsets $\mathbf{A}$ of size k, only values for subsets of size $k-1$ are required. Therefore, by computing simultaneously these two functions, when values for subsets of a given size have been computed, the memory used for smaller set can be reused. However, to be able to access G^*_w, we should redefine M_l to store optimal graphs instead of optimal sinks. The worst memory usage corresponds to $k = \lfloor \frac{n}{2} \rfloor + 1$ when we have to consider approximately $O(\frac{2^n}{\sqrt{n}})$ sets: this approximation comes from Stirling formula applied to the binomial coefficient of n and $\lfloor \frac{n}{2} \rfloor$ ($\lfloor x \rfloor$ is the highest integer less than or equal to x). At that time, $O(\sqrt{n}2^n)$ best parent sets are stored by F, and $O(\frac{2^n}{\sqrt{n}})$ graphs by M. Since a parent set requires $O(n)$ space and a graph $O(n^2)$, we derive that the maximal memory usage with recycling is $O(n^{\frac{3}{2}}2^n)$, while total memory usage of F in Algorithm 1 was $O(n^2 2^n)$. Actually, since Algorithm 1 is feasible only for small n, we can consider that a set requires $O(1)$ space (represented by less than k integers on a x-bit CPU if $n < kx$): in that case also, the memory storage is divided by a factor $\sqrt{n}$ with recycling.

Ott et al. (2005) also adapted their algorithm to list as many suboptimal graphs as desired. Such capacity is precious in order to find which structural similarities are shared by highly probable graphs, particularly when the score criteria used is not equivalent. However, for an equivalent score, since the listed graphs will be mainly on the same equivalent classes, they will probably not bring more information than the CPDAG of an optimal graph.

4. Super-Structure Constrained Optimal Search

Compare to a brute force algorithm that would browse all search space, OS achieved a considerable improvement. Graphs of around thirty nodes are still hardly computed, and many small real networks such as the classical ALARM network (Beinlich et al., 1989) with 37 variables are not feasible at all. The question of an optimal algorithm with a lower complexity is still open. In our case, we focus on structural constraint to reduce the search space and develop a faster algorithm.

4.1 Super-Structure

To keep the property that the result of OS converges to the true graph in the sample limit, the constraint should at least authorize the true skeleton. Since knowing the true skeleton is a strong assumption and learning it with high confidence from finite data is a hard task, we propose to consider a more flexible constraint than fixing the skeleton. To this end, we introduce a super-structure as:

Definition 4. *An undirected graph $S = (\mathbf{V}, \mathbf{E}_S)$ is said to be a* super-structure *of a DAG $G = (\mathbf{V}, \mathbf{E}_G)$, if the skeleton of G, $G' = (\mathbf{V}, \mathbf{E}_{G'})$ is a subgraph of S (i.e., $\mathbf{E}_{G'} \subseteq \mathbf{E}_S$). We say that S* contains *the skeleton of G.*

Considering a structure learning task, a super-structure S is said to be true or *sound* if it contains the true skeleton; otherwise it is said *incomplete*. Finally we propose to study the problem of model

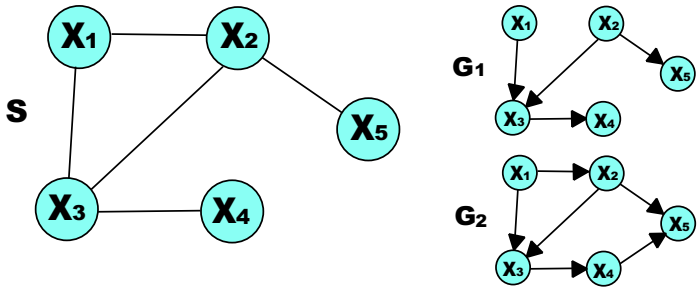


Figure 1: In a search constrained by S, G_1 could be considered but not G_2 because $\langle X_4, X_5 \rangle \notin \mathbf{E}_S$.

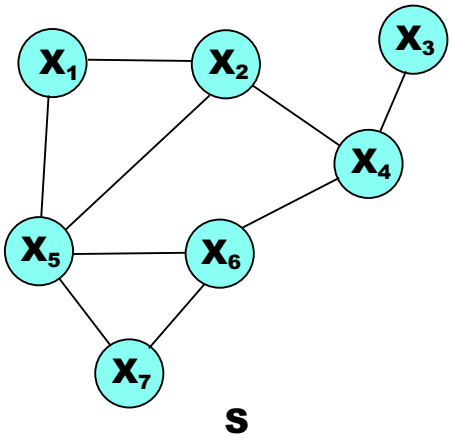
inference from data given a super-structure S: S is assumed to be sound, and the search space is restricted to DAGs whose skeletons are contained in S as illustrated in Figure 1. Actually, the "skeleton" learned by MMPC is used as a super-structure in MMHC. In fact, the skeleton of the graph returned by MMHC is not proven to be the same than the learned one; some edges can be missing. It is the same for the candidate parents in SC. Thus, the idea of super-structure already existed, but we define it explicitly, which has several advantages.

First, a dedicated terminology enables to emphasize two successive and independent phases in structure learning problem: on one hand, learning with high probability a sound super-structure S (sparse if possible); on the other hand, given such structure, searching efficiently the restricted space and returning the best optimum found (global optimum if possible). This problem cutting enables to make clearer the role and effect of each part. For example, since SC and MMHC use the same search, comparing their results allow us directly to evaluate their super-structure learning approach. Moreover, while conceiving a search strategy, it could be of a great use to consider a super-structure given. This way, instead of starting from a general intractable case, we have some framework to assist reasoning: we give some possible directions in our future work. Finally, this manner to apprehend the problem already integrates the idea that the true skeleton will not be given by an IT approach; hence, it could be better to learn a bit denser super-structure to reduce missing edges, which should improve accuracy.

Finally, we should explain how practically a sound super-structure S can be inferred. Even without knowledge about causality, a quick analysis of the system could generate a rough draft by determining which interactions are impossible; localization, nature or temporality of variables often forbid evidently many edges. In addition, for any IT-based technique to learn the skeleton, the neighborhood of variables or their Markov blanket could be used to get a super-structure. This one should become sound while increasing the significance level of the tests: this is because we only need to reduce false negative discovery. Although the method used in PC algorithm could be a good candidate to learn a not sparse but sound super-structure, we illustrate our idea with MMPC in Section 5.1.

4.2 Constraining and Optimizing

From now on, we will assume that we are given a super-structure $S = (\mathbf{X}, \mathbf{E}_S)$ over $\mathbf{X}$. We refer to the neighborhood of a variable X_i in S by $\mathbf{N}(X_i)$, that is, the set of nodes connected to X_i in S (i.e., $\{X_j \mid \langle X_i, X_j \rangle \in \mathbf{E}_S\}$); m is the maximal degree of S, that is, $m = \max_{X_i \in \mathbf{X}} |\mathbf{N}(X_i)|$. Our task is to globally maximize the score function over our reduced search space.



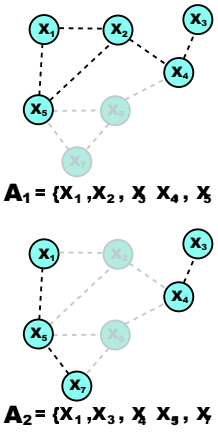


Figure 2: $\mathbf{A}_1$ is in $Con(S)$, but not $\mathbf{A}_2$ because in $S_{\mathbf{A}_2}$, X_7 and X_4 are not connected.

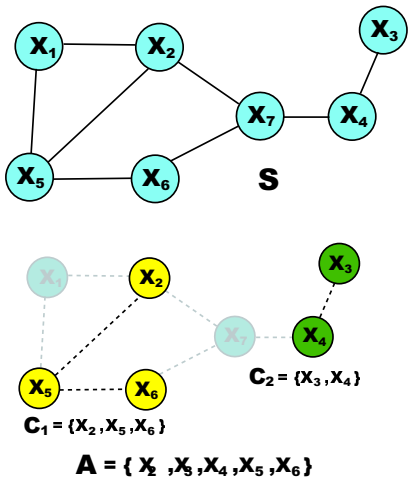


Figure 3: The maximal connected subsets of $\mathbf{A}$: $\mathbf{C}_1$ and $\mathbf{C}_2$.

Since the parents of every X_i are constrained to be included in $\mathbf{N}(X_i)$, the function F has to be defined only for $\forall \mathbf{A} \subseteq \mathbf{N}(X_i)$. Consequently, computation of F in step (b) becomes:

(b*) For each $X_i \in \mathbf{X}$ and each $\mathbf{A} \subseteq \mathbf{N}(X_i)$
Calculate $F_s(X_i, \mathbf{A})$ and $F_p(X_i, \mathbf{A})$ using (2)

Only the underlined part has been modified; clearly, F still can be computed recursively since $\forall X_j \in \mathbf{A}$, the subset $\mathbf{A} \backslash \{X_j\}$ is also included in $\mathbf{N}(X_i)$, and its F value is already known. With this slight modification, the time complexity of computing F becomes $O(n2^m)$, which is a decisive improvement opening many perspectives; more details are given at the end of this section. However, to keep formulae (5) and (3) correct, $F(X_i, \mathbf{A})$ for any subset $\mathbf{A}$ has to be replaced by $F(X_i, \mathbf{A} \cap \mathbf{N}(X_i))$. Before simplifying the calculation of M, it is necessary to introduce the notion of *connectivity*:

Definition 5. *Given an undirected graph $S = (\mathbf{X}, \mathbf{E}_S)$, a subset $\mathbf{A} \subseteq \mathbf{X}$ is said to be* connected *if $\mathbf{A} \neq \emptyset$ and the subgraph of S induced by $\mathbf{A}$, $S_{\mathbf{A}}$, is connected (cf. Figure 2).*

In our study, connectivity will always refer to the connectivity in the super-structure S. $Con(S)$ will refer to the set of connected subsets of $\mathbf{X}$. In addition, each not empty subset of $\mathbf{X}$ can be broken down uniquely into the following family of connected subsets:

Definition 6. *Given an undirected graph $S = (\mathbf{X}, \mathbf{E}_S)$ and a subset $\mathbf{A} \subseteq \mathbf{X}$, let $S_1 = (\mathbf{C}_1, \mathbf{E}_1), \cdots, S_p = (\mathbf{C}_p, \mathbf{E}_p)$ be the connected components of the induced subgraph $S_{\mathbf{A}}$. The subsets $\mathbf{C}_1, \cdots, \mathbf{C}_p$ are called the* maximal connected subsets *of $\mathbf{A}$ (cf. Figure 3).*

The most important property of the maximal connected subsets $\mathbf{C}_1, \cdots, \mathbf{C}_p$ of a subset $\mathbf{A}$ is that, when $p > 1$ (i.e., when $\mathbf{A} \notin Con(S)$) for any pair $\mathbf{C}_i$, $\mathbf{C}_j$ with $i \neq j$, $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ and there is no edges in S between nodes of $\mathbf{C}_i$ and nodes of $\mathbf{C}_j$. Next we show that the value of M for subsets that are unconnected do not have to be explicitly calculated, which is the second and last modification

of Algorithm 1. The validity of our algorithm is simultaneously proved.

Theorem 2. *A constrained optimal graph can be found by computing M only over Con*(S).
Proof: First, let consider a subset $\mathbf{A} \notin Con(S)$, its maximal connected subsets $\mathbf{C}_1, \cdots, \mathbf{C}_p$ ($p > 1$), and an optimal constrained DAG $G^* = (\mathbf{A}, \mathbf{E}^*)$. Since G^* is constrained by the super-structure, and following the definition of the maximal connected subsets, there cannot be edges in G^* between any element in $\mathbf{C}_i$ and any element in $\mathbf{C}_j$ if $i \neq j$. Therefore, the edges of G^* can be divided in p sets $\mathbf{E} = \mathbf{E}_1 \cup \cdots \cup \mathbf{E}_p$ with $G_i = (\mathbf{C}_i, \mathbf{E}_i)$ a DAG over every $\mathbf{C}_i$. Moreover, all G_i are optimal constrained graphs otherwise G^* would not be. Consequently, we can derive the two following formulas:

$$M_s(\mathbf{A}) = \sum_{i=1}^{p} M_s(\mathbf{C}_i), \tag{7}$$

$$M_l(\mathbf{A}) = M_l(\mathbf{C}_1). \tag{8}$$

Formula (7) directly follows our previous considerations that maximizing the score over $\mathbf{A}$ is equivalent to maximizing it over each $\mathbf{C}_i$ independently, since they cannot affect each other. Actually, any $M_l(\mathbf{C}_i)$ is an optimal sink and could be selected in (8); we chose $M_l(\mathbf{C}_1)$ since it is accessed faster when using the data structure proposed in Section 4.3 for M. By using (7) and (8) the value of M for unconnected subsets can be directly computed if needed from the values of smaller connected subsets. Therefore, we propose to compute M only for connected subsets by replacing step (d) with (d*) in Algorithm 2 described below. Since each singleton $\{X_i\}$ is connected, step (c) is not raising a problem. In step (d*) we consider $\mathbf{A} \in Con(S)$ and apply formula (5), if there is $X_j \in \mathbf{A}$ such that $\mathbf{B}_j = \mathbf{A} \backslash \{X_j\}$ is not connected, we then can directly calculate $M_s(\mathbf{B}_j)$ by applying (7). the values of M_s for the maximal connected subsets of $\mathbf{B}_j$ are already computed since these subsets are of smaller sizes than $\mathbf{A}$. Therefore, $M_l(\mathbf{A})$ and $M_s(\mathbf{A})$ can be computed. Finally, it is also possible to retrieve w^* from (6) by using (8) if $\mathbf{T}$ is not connected, which conclude the proof of this Theorem. ■

We can now formulate our optimized version of Algorithm 1 for optimal DAG search conditioned by a super-structure S:

Algorithm 2.

(a*) Initialize $\forall X_i \in \mathbf{X}$, $F_s(X_i, \emptyset)$ and $F_p(X_i, \emptyset)$ with (1)

(b*) For each $X_i \in \mathbf{X}$ and each $\mathbf{A} \subseteq \mathbf{N}(X_i)$
Calculate $F_s(X_i, \mathbf{A})$ and $F_p(X_i, \mathbf{A})$ using (2)

(c*) Initialize $\forall X_i$, $M_s(\{X_i\})$ and $M_l(\{X_i\})$ using (4)

(d*) For each $\mathbf{A} \in Con(S)$ with $|\mathbf{A}| > 1$
Calculate $M_s(\mathbf{A})$ and $M_l(\mathbf{A})$ using (5) and (7)

(e*) Build an optimal ordering w^* using (6) and (8)

(f*) Return the best w^*-linear graph $G^*_{w^*}$ using (3)

The underlined parts of Algorithm 2 are the modifications introduced in Algorithm 1. Computational complexity and correctness of (b*) has already been presented. With Theorem 2, validity of our algorithm is assured and since in (c*) and (d*) every element of $Con(S)$ are considered only

once, the total computational complexity is in $O(n2^m + |Con(S)|)$; here again complexity refers to the number of times formulae (2) or (5) are computed. We will describe in the next Section a method to consider only connected subsets, and come over the number of connected subsets of S in Section 4.4. Although set operators are used heavily in our algorithm, such operations can be efficiently implemented and considered of a negligible cost as compared to other operations, such as score calculations. Concerning the complexity of calculating F, $O(n2^m)$ is in fact a large upper bound. Still, since it depends only linearly on the size of the graphs, F can be computed for graphs of any size if their maximal degree is less than around thirty. This enables usage of this function in new search strategies for many real systems that could not be considered without constraint. We should remark that some cases of interest still cannot be studied since this upper limitation on m constrains also the maximal number of children of every variables. However, this difficulty concerns also many IT-approaches since their complexity also depends exponentially on m (see Tsamardinos et al. 2006 for MMPC and Kalisch and Bühlmann 2007 for PC). Finally, like in Algorithm 1, the number of scores calculated can be reduced to $O(nm^c)$ by constraining on the number of parents.

Although the number of M and F values calculated is strictly reduced, a potential drawback of Algorithm 2 is that memory cannot be recycled anymore. First, when (5) is used during step (d*), now $F_s(X_i, \mathbf{B}_j \cap \mathbf{N}(X_i))$ is required, and nothing can be said about $|\mathbf{B}_j \cap \mathbf{N}(X_i)|$ implying that we should store every value of F_s computed before. Similar arguments hold for F_p in case M_l is used to store optimal graphs, and for M_s and M_l because (7) and (8) could have to be used anytime during (d*) and (e*) respectively. However, since space complexity of F is $O(n2^m)$ and the one of M is $O(|Con(S)|)$ (cf. next section), if m is bounded Algorithm 2 should use less memory than Algorithm 1 even when recycling memory (i.e., $O(\sqrt{n}2^n)$ assuming that a set takes $O(1)$ space). This soften the significance of recycling memory in our case.

Finally, since our presentation of Algorithm 2 is mainly formal, we should detail how it is practically possible to browse efficiently only the connected subsets of $\mathbf{X}$. For this, we present in the next section a simple data structure to store values of M and a method to build it.

4.3 Representation of $Con(S)$

For every $\mathbf{A} \in Con(S)$ we define $\mathbf{N}(\mathbf{A}) = \left(\bigcup_{X_i \in \mathbf{A}} \mathbf{N}(X_i)\right) \setminus \mathbf{A}$, that is the set of variables neighboring $\mathbf{A}$ in S. For every $X_i \notin \mathbf{A}$, we note $\mathbf{A}_i^+ = \mathbf{A} \cup \{X_i\}$, it is connected if and only if $X_i \in \mathbf{N}(\mathbf{A})$. Finally, for a subset $\mathbf{A}$ not empty, let $min(\mathbf{A})$ be the smallest index in $\mathbf{A}$, that is, $min(\mathbf{A}) = i$ means that $X_i \in \mathbf{A}$ and $\forall X_j \in \mathbf{A}$, $j \geq i$; by convention, $min(\emptyset) = 0$. Now we introduce an auxiliary directed graph $G^\star = (Con(S)^\star, \mathbf{E}^\star)$, where $Con(S)^\star = Con(S) \cup \{\emptyset\}$ and the set of directed edges $\mathbf{E}^\star$ is such that there is an edge from $\mathbf{A}$ to $\mathbf{B}$ if and only if $\mathbf{A} \subset \mathbf{B}$ and $|\mathbf{B}| = |\mathbf{A}| + 1$. In other words, with the convention that $\mathbf{N}(\emptyset) = \mathbf{X}$, $\mathbf{E}^\star = \{(\mathbf{A}, \mathbf{A}_i^+), \forall \mathbf{A} \in Con(S)^\star \text{ and } \forall X_i \in \mathbf{N}(\mathbf{A})\}$. Actually, $G^\star$ is trivially a DAG since arcs always go from smaller to bigger subsets. Finally, let define H as being the spanning tree obtained from the following depth-first-search (DFS) on $G^\star$:

- The search starts from $\emptyset$.
- While visiting $\mathbf{A} \in Con(S)^\star$: for all $X_{k_i} \in \mathbf{N}(\mathbf{A})$ considered by increasing indices (i.e., such that $k_1 < \cdots < k_p$, where $p = |\mathbf{N}(\mathbf{A})|$) visit the child $\mathbf{A}_{k_i}^+$ if it was not yet done. When all children are done, the search backtracks.

Since there is a path from the empty set to every connected subset in $G^\star$, the nodes of the tree H represent unambiguously $Con(S)^\star$. We use H as a data structure to store the values of M in

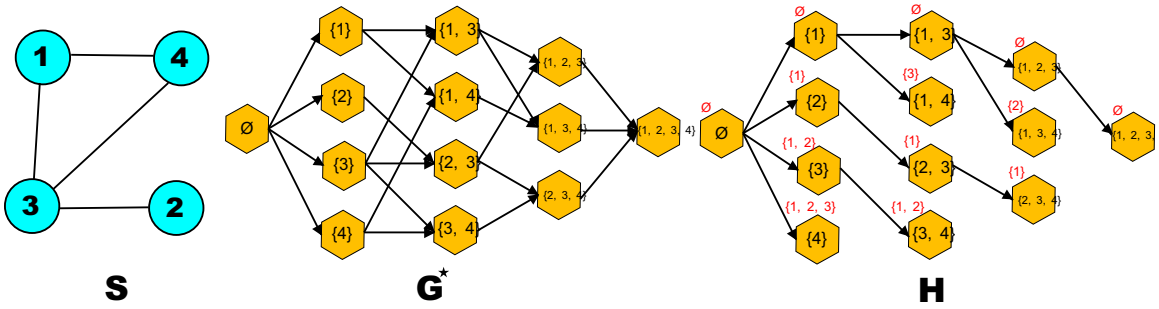


Figure 4: $G^\star$ and H for a given S. $\mathbf{Fb}(\mathbf{A})$ is indicated in red above each node in H.

Algorithm 2; this structure is illustrated by an example in Figure 4. Further, we propose a method to build H directly from S without having to build $G^\star$ explicitly. First, we notice that after visiting $\mathbf{A}$, every $\mathbf{B} \in Con(S)$ such that $\mathbf{B} \supset \mathbf{A}$ have been visited for sure. When visiting $\mathbf{A}$, we should consider only the children of $\mathbf{A}$ that were not yet visited. For this, we define:

- $\mathbf{Fb}(\mathbf{A})$ is the set of forbidden variables of $\mathbf{A}$, that is: for every $\mathbf{B} \in Con(S)$ with $\mathbf{B} \supset \mathbf{A}$, $\mathbf{B}$ has been already visited if and only if $\mathbf{B} \supseteq \mathbf{A}^+_j$ with $X_j \in \mathbf{Fb}(\mathbf{A})$.

By defining recursively this forbidden set for every child of $\mathbf{A}$ that has not yet been visited, we derive the following method to build H:

Method 1.
1: Create the root of H (i.e., $\emptyset$), and initialize $\mathbf{Fb}(\emptyset) = \emptyset$.
2: For i from 0 to $n-1$, and for all $\mathbf{A}$ in H such that $|\mathbf{A}| = i$
3: Set $\mathbf{Fb}^* = \mathbf{Fb}(\mathbf{A})$,
4: For every $X_{k_j} \in \mathbf{N}(\mathbf{A}) \backslash \mathbf{Fb}(\mathbf{A})$ considered by increasing indices
5: Add to $\mathbf{A}$ the child $\mathbf{A}^+_{k_j}$ in H and define $\mathbf{Fb}(\mathbf{A}^+_{k_j}) = \mathbf{Fb}^*$
6: Update $\mathbf{Fb}^* = \mathbf{Fb}^{*+}_{k_j}$

The correctness of Method 1 is proven in the next Theorem. In order to derive the time complexity of step (d*) in terms of basic operations, while using Method 1 in Algorithm 2, let consider that the calculation takes place on a x-bit machine and that n is at maximum few times greater than x. Thus, subsets requires $O(1)$ space, and any operations on subsets are done in $O(1)$ time, except $min(\mathbf{A})$ (in $O(\log(n))$ time).

Theorem 3. *The function M can be computed in time and space proportional to $O(|Con(S)|)$, up to some polynomial factors. With Method 1, M is computed in $O(\log(n)n^2|Con(S)|)$ time and requires $O(|Con(S)|)$ space.*

Proof: First, to prove correctness of Method 1, we show that if $\mathbf{Fb}(\mathbf{A})$ is correctly defined in regard to our DFS, the search from $\mathbf{A}$ proceeds as expected and that before back tracking every connected superset of $\mathbf{A}$ has been visited. The case when all the variables neighboring $\mathbf{A}$ are forbidden being trivial, we directly consider all the elements $X_{k_1}, \cdots, X_{k_p}$ of $\mathbf{N}(\mathbf{A}) \backslash \mathbf{Fb}(\mathbf{A})$ by increasing indices like in DFS (with $p \geq 1$). Then, $\mathbf{A}^+_{k_1}$ should be visited first, and $\mathbf{Fb}(\mathbf{A}^+_{k_1}) = \mathbf{Fb}(\mathbf{A})$ because $\forall X_j \in$

$\mathbf{Fb}(\mathbf{A})$ among the supersets of $\mathbf{A}_j^+$ there is also the supersets of $\mathbf{A}_{k_1,j}^{++}$. Now let suppose that the forbidden sets were correctly defined and that the visits correctly proceeded until $\mathbf{A}_{k_i}^+$: if $i = p$ by hypothesis we explored every connected supersets of the children of $\mathbf{A}$ and the search back track correctly. Otherwise, we should define $\mathbf{Fb}(\mathbf{A}_{k_{i+1}}^+) = \mathbf{Fb}(\mathbf{A}) \cup \{X_{k_1}\} \cup \cdots \cup \{X_{k_i}\} = \mathbf{Fb}(\mathbf{A}_{k_i}^+)_{k_i}^+$ to take into account all supersets visited during the previous recursive searches. Although Method 1 does not proceed recursively (to follow the definition of M), since it uses the same formulae to define the forbidden sets, and since $\mathbf{Fb}(\emptyset)$ is correctly initialized, H is built as expected.

To be able to access easily $M(\mathbf{A})$, we keep for every node an auxiliary set defined by $\mathbf{Nb}(\mathbf{A}) = \mathbf{N}(\mathbf{A}) \setminus \mathbf{Fb}(\mathbf{A})$ that is easily computed as processing Method 1. Since there is $O(|Con(S)|)$ nodes in H, each storing a value of M and two subsets requiring $O(1)$ space, the assertion about space complexity is correct.

Finally, building a node requires $O(1)$ set operations. To access $M(\mathbf{A})$ even for an unconnected subset, we proceed on the following manner: we start from the root, and define $\mathbf{T} = \mathbf{A}$. Then, when we rushed the node $\mathbf{B} \subseteq \mathbf{A}$, with $i = min(\mathbf{Nb}(\mathbf{B}) \cap \mathbf{T})$, we withdraw X_i from $\mathbf{T}$, and go down to the i^{th} child of $\mathbf{B}$. If $i = 0$ then if $\mathbf{T} = \emptyset$ we found the node of $\mathbf{A}$; otherwise we rushed the first maximal connected component of $\mathbf{A}$, that is, $\mathbf{C}_1$ of which we can accumulate the M value in order to apply (7) or (8). In that case, we continue the search by restarting from the root with the variables remaining in $\mathbf{T}$. In any cases, at maximum *min* is used $O(n)$ times to find $M(\mathbf{A})$, implying a time complexity of $O(\log(n)n^2)$ for formula (5). ■

It is interesting to notice that, even without memoization, the values of M can be calculated in different order. For example, by calling H_i the subtree of H starting from $\{X_i\}$, only values of M over H_j such that $j \geq i$ are required. Then it is feasible to calculate M from H_n to H_1, which could be used to apply some branch-pruning rules based on known values of M or to apply different strategies depending on the topology of the connected subset; these are only suppositions.

More practically, other approach could be proposed to build a spanning tree of $G^\star$ and Method 1 is presented here only to complete our implementation and illustrate Theorem 3. We should note that one can also implement the calculation of M over $Con(S)$ in a top-down fashion using memoization and an auxiliary function to list maximal connected components of a subset. However, such implementations will be less efficient both in space and time complexity. Even without considering the cost of recursive programming, listing connected components is in $O(nm)$. Then, in order to not waste an exponential factor of memory, self balanced trees should be used to store the memorized values of M: it would require $O(n)$ time to access a value and up to $O(n^2)$ if (7) is used. This should be repeated $O(n)$ times to apply (5), which implies a complexity of around $O(n^3|Con(S)|)$. Consequently, we believe that Method 1 is a competitive implementation of step (d*).

4.4 Counting Connected Subsets of a Graph

To understand the complexity of Algorithm 2, the asymptotic behavior of $|Con(S)|$ should be derived, depending on some attributes of S. Comparing the trivial cases of a linear graph (i.e., $m \leq 2$) where $|Con(S)| = O(n^2)$ and a star graph (i.e., one node is the neighbor of all others) where $|Con(S)| = 2^{n-1} + n - 1$ clearly indicates that $|Con(S)|$ depends strongly on the degrees of Srather than on the number of edges or number of cycles. One important result from Björklund et al. (2008) is that $|Con(S)| = O(\beta_m^n)$ with $\beta_m = (2^{m+1} - 1)^{\frac{1}{m+1}}$ a coefficient that only depends on the maximal degree m of S.

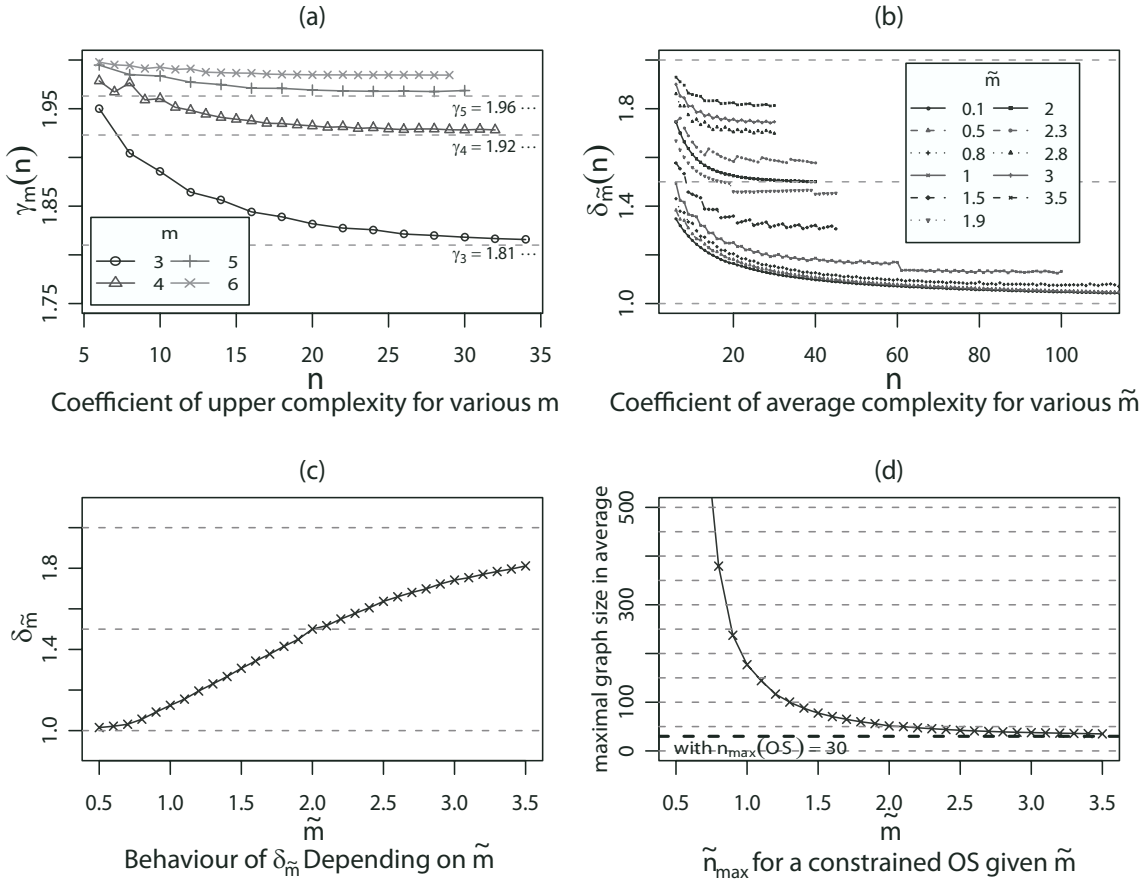


Figure 5: Experimental derivation of γ_m, $\delta_{\tilde{m}}$ and $\tilde{n}_{\max_2}(\tilde{m})$.

Still, since this upper bound is probably over-estimated, we tried to evaluate a better one experimentally. For every pair (m,n) of parameters considered , we randomly generated 500 undirected graphs S pushing their structures towards a maximization of the number of connected subsets. For this, all the nodes had exactly m neighbors and S should at least be connected. Then, since after a first series of experiments the most complex structures appeared to be similar to full $(m-1)$-trees, with a root having m children and leaves being connected to each other, only such structures were considered during a second series of experiments. Finally, for each pair (m,n), from the maximal number $R_{n,m}$ of connected subsets found, we calculated $\exp(\frac{\ln(R_{n,m})}{n})$ in order to search for an exponential behavior as shown in Figure 5(a).

Results 1. *Our experimental measures led us to propose that* $|Con(S)| = O({\gamma_m}^n)$ *(cf. Figure 5(a)).*

The weak point of our strategy is that more graphs should be consider while increasing n, since the number of possible graphs also increases. Unfortunately, this is hardly feasible in practice since counting gets longer with larger graphs. Nevertheless, Results 1 were confirmed during a more detailed study of the case $m = 3$ using 10 times more graphs and up to $n = 30$. In addition, even this estimated upper bound is practically of a limited interest since it still dramatically overestimates $|Con(S)|$ for real networks. Real networks are not, in general, as regular and as dense graphs as

Complexity	Algorithm 1 (OS)	Algorithm 2 (COS)
Time (steps)	$O(n2^n)$	$O(\|Con(S)\|)$
in details	$O(n^2 2^n)$	$O(\log(n)n^2\|Con(S)\|)$
scores computed	$n2^n$	$O(n2^m)$
if $\|\mathbf{Pa}_i\| < c$	$O(n^{c+1})$	$O(nm^c)$
Space	$O(\sqrt{(}n)2^n)$	$O(\|Con(S)\|)$

Table 1: Improvement achieved by COS.

S	$\|Con(S)\|$	some values
Tree-like	$O(\alpha_m^n)$	$\alpha_3 \approx 1.58$, $\alpha_4 \approx 1.65$, $\alpha_5 \approx 1.707$
General	$O(\beta_m^n)$	$\beta_3 \approx 1.968$, $\beta_4 \approx 1.987$, $\beta_5 \approx 1.995$
Measured	$O(\gamma_m^n)$	$\gamma_3 \approx 1.81$, $\gamma_4 \approx 1.92$, $\gamma_5 \approx 1.96$
In average	$O(\delta_{\tilde{m}}^n)$	$\delta_{1.5} \approx 1.3$, $\delta_2 \approx 1.5$, $\delta_{2.5} \approx 1.63$, $\delta_3 \approx 1.74$

Table 2: Results on $|Con(S)|$

the ones used in previous experiments. To illustrate that $O({\gamma_m}^n)$ is a pessimistic upper bound, we derived the theoretical upper bound of $|Con(S)|$ for tree-like structures of maximal degree m. This is given as an example, although it might help estimation of $|Con(S)|$ for structures having a bounded number of cycles. The proof is deferred to the Appendix.

Proposition 1. *If S is a forest, then $|Con(S)| = O(\alpha_m^n)$ with $\alpha_m = (\frac{2^{m-1}+1}{2})^{\frac{1}{m-1}}$.*

Finally, we studied the average size of $|Con(S)|$ for a large range of average degrees $\tilde{m}$. For each pair $(n,\tilde{m})$ considered, we generated 10000 random graphs and averaged the number of connected subsets to obtain $R_{n,\tilde{m}}$. No constraint was imposed on m, since graphs were generated by randomly adding edges until $\lfloor \frac{n\tilde{m}}{2} \rfloor$. In each case, we calculated $\exp(\frac{\ln(R_{n,\tilde{m}})}{n}))$ to search for an asymptotic behavior on $\tilde{m}$.

Results 2. *On average, for super-structures having an average degree of $\tilde{m}$, $|Con(S)|$ increases asymptotically as $O({\delta_{\tilde{m}}}^n)$, see Figure 5(b) and (c) for more details.*

Based on the assumption that Algorithm 1 is feasible at maximum for graphs of size $n_{\max_1} = 30$ (Silander and Myllymäki, 2006), we calculated $\tilde{n}_{\max_2}(\tilde{m}) = n_{\max_1}\frac{\ln(2)}{\ln(\delta_{\tilde{m}})}$ that can be interpreted as an estimation of the maximal size of graphs that can be considered by Algorithm 2 depending on $\tilde{m}$. As shown in Figure 5(d), on average, it should be feasible to consider graphs having up to 50 nodes with Algorithm 2 if $\tilde{m} = 2$. Moreover, since $\lim_{\tilde{m}\to 0} \delta_{\tilde{m}} = 1$, our algorithm can be applied to graphs of any sizes if they are enough sparse. Unfortunately, the case when $\tilde{m} < 2$ is not really interesting since it implies that networks are mainly unconnected.

To conclude, we summarize and compare the time and space complexities of both Algorithms in Table 1, using the same hypothesis on the size of a subset as Section 4.3. We neglected the complexity due to F in our algorithm, which is justified if m is not huge. Concerning the space complexity of Algorithm 1, the maximal space needed while recycling memory is used. Results on $O(|Con(S)|)$ are listed in Table 2. If super-structures S are relatively sparse and have a bounded maximal degree, the speed improvement of Algorithm 2 over Algorithm 1 should increase expo-

nentially with n. Moreover, our algorithm can be applied to some small real networks that are not feasible for Algorithm 1.

5. Experimental Evaluation

Although the demonstrations concerning correctness and complexity of Algorithm 2 enable us to anticipate results obtained experimentally, some essential points remain to be studied. Among them, we should demonstrate practically that, in the absence of prior knowledge, it is feasible to learn a sound super-structure with a relaxed IT approach. We choose to test our proposition on MMPC (Tsamardinos et al., 2006) in Section 5.2; the details of this algorithm are briefly reintroduced in Section 5.1.4. Secondly, we compare COS to OS to confirm the speed improvement of our method and study the effect of using a sound constraint in Section 5.3. We also should evaluate the worsening in terms of accuracy due to the incompleteness of an approximated super-structure. In this case we propose and evaluate a greedy method, COS+, to improve substantially the results of COS. In Section 5.4, we compare our methods to other greedy algorithms to show that, even with an incomplete super-structure, COS, and especially COS+, are competitive algorithms to study small networks. Finally we illustrate this point by studying the ALARM Network (Beinlich et al., 1989) in Section 5.5, a benchmark of structure learning algorithm for which OS is not feasible.

5.1 Experimental Approach

Except in the last real experiment, we are interested in comparing methods or algorithms for various set of parameters, such as: the size of the networks n, their average degree $\tilde{m}$ (to measure effect of sparseness on Algorithm 2), the size of data considered d and the significance level α used to learn the super-structure.

5.1.1 NETWORKS AND DATA CONSIDERED

Due to the size limitation imposed by Algorithm 1, only small networks can be learned. Further, since there it is hardly feasible to find many real networks for every pair $(n,\tilde{m})$ of interest, we randomly generated the networks to which we apply structure learning. Given a pair $(n,\tilde{m})$, DAGs of size n are generated by adding randomly $\lfloor \frac{n\tilde{m}}{2} \rfloor$ edges while assuring that cycles are not created.

For simplicity, we considered only Boolean variables; therefore, Bayesian networks are obtained from each DAG by generating conditional probabilities $P(X_i = 0|\mathbf{Pa}_i = \mathbf{pa}_i^k)$ for all X_i and all possible $\mathbf{pa}_i^k$ by choosing a random real number in $]0,1[$. Then, d data are artificially generated from such Bayesian networks, by following their entailed probability distribution.

Finally, the data are used to learn a network with every algorithm and some criteria are measured. In order to generalize our results, we repeat g times the previous steps for each quadruplet $(n,\tilde{m},\alpha,d)$. The values of each criterion of comparison for every algorithm are averaged on the g learned graphs.

5.1.2 COMPARISON CRITERIA

While learning Bayesian networks, we evaluate the performances of every algorithm on three criteria. Since the learning task consists in the maximization of the score function, a natural criterion to evaluate the quality of the learned network is its score. In our experiments we use BIC because of its speed of evaluation. Since we are interested in comparing results in terms of score depending on

n or $\tilde{m}$ in a diagram, we do not directly represent scores (their values change radically for different parameters) but use a score ratio to the optimal score: $\frac{Score(G_{OS})}{Score(G_{\text{Other}})}$, where the label of the graph indicates which Algorithm was used. The better is the score obtained by an algorithm, the closer to 1 is its score ratio. We preferred to use the best score rather than the score of the true network, because the true network is rarely optimal; its score is even strongly penalized if its structure is dense and data sets are small. Therefore, it is not convenient to use it as a reference in terms of score.

The second criterion is a measure of the complexity estimated by the execution time of each algorithm, referred as *Time*. Of course, this is not the most reliable way to estimate complexity, but since calculations are done on the same machine, and since measures are averaged on few similar calculations, execution time should approximate correctly the complexity. To avoid bias of this criterion, common routines are shared among algorithms (such as the score function, the structure learning method and the hill climbing search).

Finally, since our aim is to learn a true network, we use a structural hamming distance that compares the learned graph with the original one. As proposed in Tsamardinos et al. (2006), to take into consideration equivalence classes, the CPDAGs of both original and learned DAGs are built and compared. This defines the structural error ratio SER of a learned graph, which is the number of extra, missing, and wrongly oriented edges divided by the total number of edges in the original graph. In our case, we penalize wrongly oriented edges only by half, because we consider that errors in the skeleton are more "grave" than those in edges orientation. The reason is not only visual: a missing edge, or an extra edge, implies more mistakes in terms of conditional independencies in general than wrongly oriented ones. Moreover, in CPDAGs, the fact that an edge is not correctly oriented is often caused by extra or missing edges. Furthermore, such a modification does not intrinsically change the results, since it benefits every algorithm on the same manner.

5.1.3 HILL CLIMBING

Although hill climbing searches are used by different algorithms, we implemented only one search that is used in all cases. This search can consider a structural constraint S, and is given a graph G_{init} from which to start the search. Then it processes as summarized in Section 2.2, selecting at each step the best transformation among all edge withdrawals, edge reversals and edge additions according to the structure constraint. The search stops as soons as the score cannot be strictly improved anymore. If several transformations involve the same increase of the score, the first transformation encountered is applyed. This implies that the results will depend on the ordering of the variables; however, since the graphs considered are randomly generated, their topological ordering is also random, and in average the results found by our search should not be biased.

5.1.4 RECALL ON MMPC

In Section 5.3 a true super-structure is given as a prior knowledge; otherwise we should use an IT-approach to approximate the structural constraint S from data. Since MMHC algorithm is included in our experiments, we decided to illustrate our idea of relaxed independency testing on MMPC strategy (Tsamardinos et al., 2006).

In MMPC, the following independency test is used to infer independencies among variables. Given two variables X_i and X_j, it is possible to measure if they are independent conditioning on a subsets of variables $\mathbf{A} \subseteq \mathbf{X} \backslash \{X_i, X_j\}$ by using the G^2 statistic (Spirtes et al., 2000), under the null hypothesis of conditional independency holding. Referring by $N^{ab\mathbf{c}}$ to the number of times that

$X_i = a$, $X_j = b$ and $\mathbf{A} = \mathbf{c}$ simultaneously in the data, G^2 is defined by:

$$G^2 = 2 \sum_{a,b,\mathbf{c}} N^{ab\mathbf{c}} \ln \left(\frac{N^{ab\mathbf{c}} N^{\mathbf{c}}}{N^{a\mathbf{c}} N^{b\mathbf{c}}} \right).$$

The G^2 statistic is asymptotically distributed as χ^2 under the null hypothesis. The χ^2 test returns a p-value, $P_{IT}(X_i, X_j|\mathbf{A})$, that corresponds to the probability of falsely rejecting the null hypothesis given it is true; in MMPC, the effective number of parameters defined in Steck and Jaakkola (2002) is used as degree of freedom. Thus given a significance level α, if $P_{IT} \leq \alpha$ null hypothesis is rejected, that is, X_i and X_j are considered as conditionally dependent. Otherwise, the hypothesis is accepted (abusing somehow of the meaning of the test), and variables are declared conditionally independent. The main idea of MMPC is: given a conditioning set $\mathbf{A}$, instead of considering only $P_{IT}(X_i, X_j|\mathbf{A})$ to decide dependency, it is more robust to consider $\max_{\mathbf{B} \subseteq \mathbf{A}} P_{IT}(X_i, X_j|\mathbf{B})$; that way, the decision is based on more tests; p-values already computed are cached and reused to calculate this maximal p-value. Finally, MMPC build the neighborhood of each variable X_i (called the set of parents and children, or **PC**) by adding successively potential neighbors of X_i from a temporary set $\mathbf{T}$. While conditioning on the actual neighborhood **PC**, the variable $X_k \in \mathbf{T}$ that minimizes the maximal p-value defined before is selected because it is the variable the most related to X_i. During this phase, every variable that appears independent of X_i is not considered anymore and withdrawn from $\mathbf{T}$. Then when X_k is added to **PC**, we test if all neighbors are always conditionally dependent: if some are not, they are withdrawn from **PC** and not considered anymore. This process ends when $\mathbf{T}$ becomes empty.

We present further the details of our implementation of MMPC, referred as Method 2; it is slightly different from the original presentation of MMPC, but the main steps of the algorithm are the same. One can prove by using Theorem 1 that if the independencies are correctly estimated, this Method should return the true skeleton, which should be the case in the sample limit. About computational complexity, one can derive that MMPC should calculate around $O(n^2 2^m)$ tests in average. However, nothing can be said in practice about the maximal size of **PC**, especially if many false dependencies occurs. Therefore, the time complexity of MMPC can be in the worst case $O(n^2 2^n)$.

Method 2 (MMPC). (Tsamardinos et al., 2006)

1: For $\forall X_i \in \mathbf{X}$
2: Initialize $\mathbf{PC} = \emptyset$ and $\mathbf{T} = \mathbf{X} \setminus \{X_i\}$
3: While $\mathbf{T} \neq \emptyset$
4: For $\forall X_j \in \mathbf{T}$, if $\max_{\mathbf{B} \subseteq \mathbf{PC}} P_{IT}(X_i, X_j|\mathbf{B}) > \alpha$ then $\mathbf{T} = \mathbf{T} \setminus \{X_j\}$
5: Define $X_k = \min_{X_j \in \mathbf{T}} \max_{\mathbf{B} \subseteq \mathbf{PC}} P_{IT}(X_i, X_j|\mathbf{B})$ and $\mathbf{PC} = \mathbf{PC} \cup \{X_k\}$
6: For $\forall X_j \in \mathbf{PC} \setminus \{X_k\}$, if $\max_{\mathbf{B} \subseteq \mathbf{PC} \setminus \{X_j\}} P_{IT}(X_i, X_j|\mathbf{B}) > \alpha$ Then $\mathbf{PC} = \mathbf{PC} \setminus \{X_j\}$
7: $\mathbf{N}(X_i) = \mathbf{PC}$
8: For $\forall X_i \in \mathbf{X}$ and $\forall X_j \in \mathbf{N}(X_i)$
9: If $X_i \notin \mathbf{N}(X_j)$ Then $\mathbf{N}(X_i) = \mathbf{N}(X_i) \setminus \{X_j\}$

5.2 Learning a Super-structure with MMPC

To emphasize the feasibility of learning a super-structure from data, we study how changes the skeleton learned by MMPC while α increases, considering various cases of $(n, \tilde{m}, d)$. As proposed

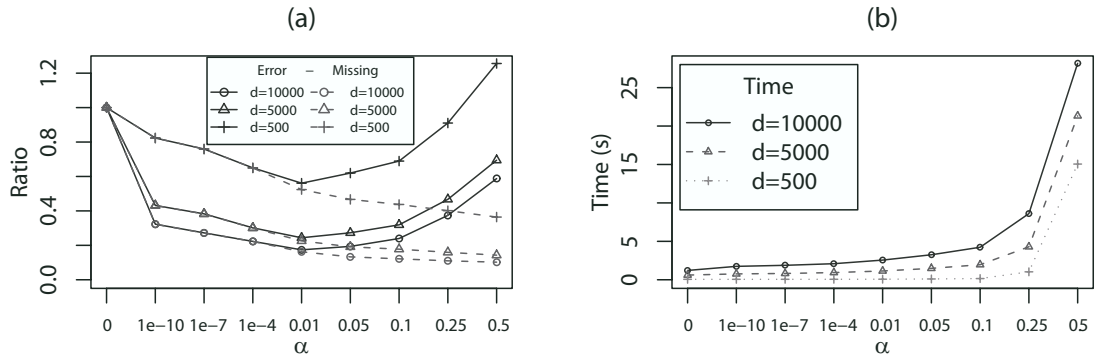


Figure 6: Effects of d and α on the results of Method 2.

before, we average our criteria of interest over $g = 50$ different graphs for every set of parameters. In the present case our criteria are: the time of execution *Time*, the ratio of wrong edges (missing and extra) $Error$, and the ratio of missing edges *Miss* of the learned skeleton. Here again, these ratios are defined while comparing to the true skeleton and dividing by its total number of edges $\lfloor \frac{n\tilde{m}}{2} \rfloor$. While learning a skeleton, $Error$ should be minimized; however in the case of learning a super-structure, *Miss* is the criterion to minimize.

Results 3 $\lim_{\alpha \to 1} Miss(\alpha) = 0$, *which validates our proposition of using higher* α *to learn super-structures (cf. Figure 6(a)). Of course, we obtain the same results with increasing data,* $\lim_{d \to \infty} Miss(\alpha) = 0$. *However, since when* $\alpha \to 1$, $Time(\alpha) \approx O(n^2 2^{n-2})$, *high values of* α *can be practically infeasible (cf. Figure 6(b)). Therefore, to escape a time-consuming structure-learning phase,* α *should be kept under* 0.25 *if using MMPC.*

In Figure 6(a), one can also notice that $Error$ is minimized for $\alpha \approx 0.01$, that is why such values are used while learning a skeleton. Next, we summarize the effect of n and $\tilde{m}$ on the criteria:

Results 4 *Increasing* α *improves uniformly the ration of missing edges independently of n and* $\tilde{m}$ *(cf. Figure 7(c) and (d)).* $Miss(\alpha)$ *is not strongly affected by increasing n but it is by increasing* $\tilde{m}$*; thus for dense graphs, the super-structures approximated by MMPC will probably not be sound.*

Previous statement could be explained by the fact that when $\tilde{m}$ increases, the size of conditional probability tables of each node increases enormously. Thus, the probability of creating weak or nearly unfaithful dependencies between a node and some of its parents also increases. Therefore, the proportion of edges that are difficult to learn increases as well. To complete analysis of Figure 7, we can notice as expected that *Time* increases on a polynomial manner with n (cf. Figure 7 (e)), which penalizes especially the usage of high α. $Error(\alpha)$ is minimized in general for $\alpha = 0.01$ or $\alpha = 0.05$ depending on n and $\tilde{m}$: for a given $\tilde{m}$, lower α (such as $\alpha = 0.01$) are better choices when n increases (cf. Figure 7(a)); conversely, if $\tilde{m}$ increases for a fixed n, higher α (such as $\alpha = 0.05$) are favored (Figure 7(b)). This can be justified, since if $\tilde{m}$ is relatively high, higher α that are more permissive in terms of dependencies will miss less edges, as opposed to lower ones with finite set of data.

In conclusion, although $\alpha \approx 0.01$ is preferable while learning a skeleton with MMPC, higher significance leves can be used to reduce the number of missing edges, and approximate a super-structure. Still, in this case, due to the exponential time complexity of MMPC if α is too high,

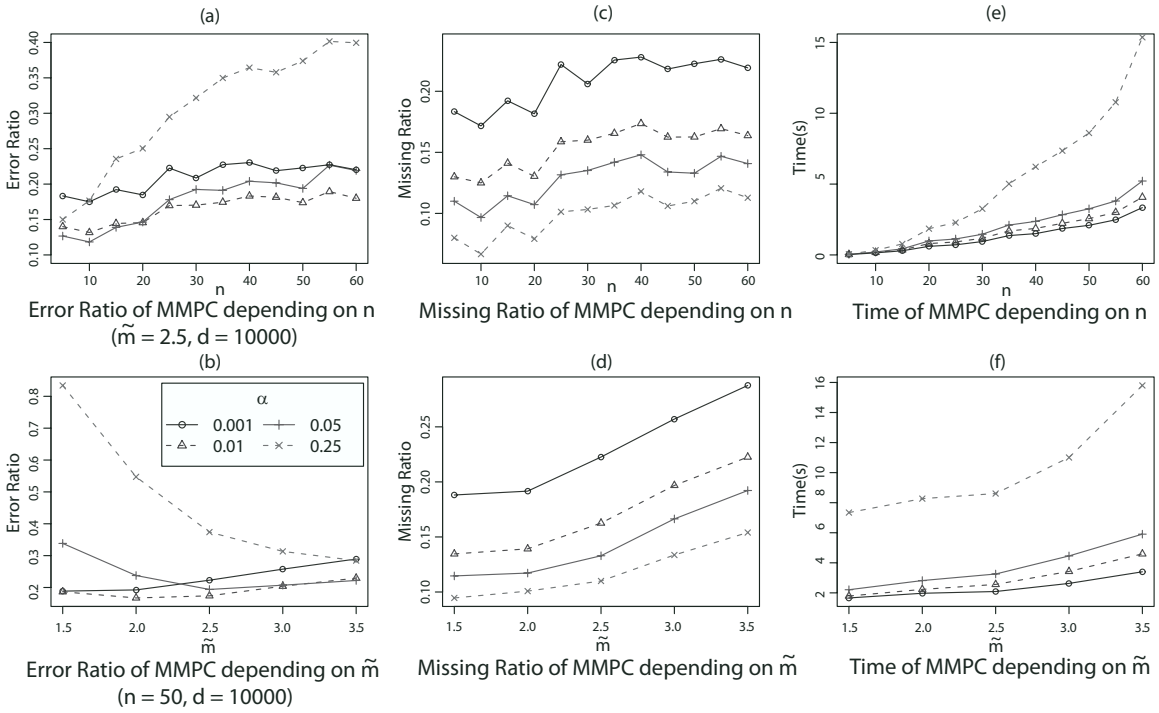


Figure 7: The criteria depending on $\tilde{m}$ and n, for α in $[0.001, 0.01, 0.05, 0.25]$.

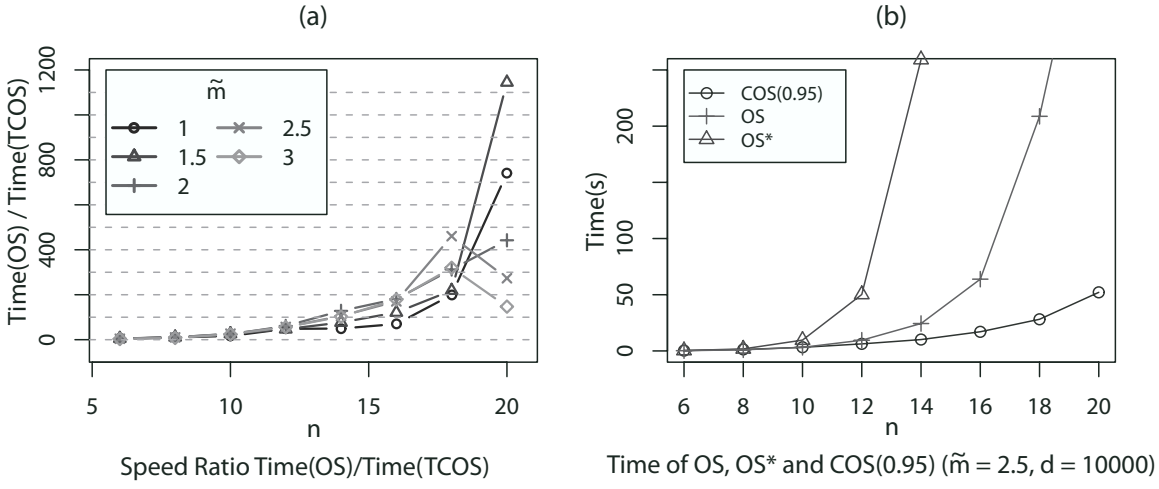


Figure 8: Comparing time complexity of OS, TCOS, and COS(α)

values of α should be selected to compromise between time complexity and ratio of missing edges. Except for $n < 30$ with $\tilde{m} < 2.5$ where $\alpha = 0.25$ is feasible and gives good results, the learned super-structure will be probably incomplete, especially if the original graph is dense: consequently, COS will not perform as well as OS when S is approximated with MMPC.

5.3 Comparison of OS and COS

In this second series of experiment, we compare Algorithm 1 and Algorithm 2 over a large sampling of graphs, for confirming and evaluating results presented in Table 1. Algorithm 1, referred as OS, is always given the true maximal number of parents c for each structure learning: this way, its execution time is considerably reduced, and we could conduct our experiments in a reasonable time. Still, to emphasize that this prior knowledge is considerably improving the speed of OS, we also considered another version of Algorithm 1 that uses in all cases a standard maximal number of parents equal to 10: it is referred as OS*. Regarding Algorithm 2, two cases are took into consideration:

- TCOS: a sound super-structure is given to the algorithm; we use the true skeleton.
- COS(α): S is learned from the data by using MMPC and a significance level α: a wide range of values are tested.

However, we know from previous section that MMPC will probably learn an incomplete super-structure, and that $G_{\mathrm{COS}(\alpha)}$ will be penalized both in its score and accuracy. Therefore, we check the effect of applying a post-processing unconstrained hill climbing search starting from $G_{\mathrm{COS}(\alpha)}$. In fact, $G_{\mathrm{COS}(\alpha)}$ might not be a local optimum if the super-structure constraint is removed; further, it could be a good starting point for a hill climbing search. The post processed version of COS is referred as COS+. Finally we compare all those algorithms for every $n \in [6,8,\cdots,20]$, $\tilde{m} \in [1,1.5,\cdots,3]$, $d \in [500,1000,\cdots,10000]$ while averaging the criteria of interest over $g = 30$ graphs: in total 4800 random graphs were used. Since there was no relevant differences depending on d, only the results for $d = 10000$ are reported here.

Results 5 *As expected, TCOS and COS(α) proceed exponentially faster than OS, even with $\alpha \approx 1$ (cf. Figures 8 (a) and (b)).*

Interestingly, the speed factor in Figure 8(a), is not purely exponential, especially for higher $\tilde{m}$. This is because, the complexity of OS is due to the costly $O(n^c)$ score calculations (here $c = O(\tilde{m})$) and the $O(2^n)$ steps (b) and (d) that dominate the complexity only for large n. As for TCOS, it only calculates $O(n)$ scores and needs around $O(\delta_{\tilde{m}}^n)$ steps. Thus, the speed factor starts by increasing fast with n because of the $O(n^c)$ scores of OS, before decreasing to behave asymptotically as $O(\left(\frac{2}{\delta_{\tilde{m}}}\right)^n)$. In Figure 8(b), the complexity of OS* is also represented: without limitation on the number of parents, the speed ratio would be purely exponential.

To evaluate the quality of the graphs learned depending on the algorithm used, for $n = 12$ and $\tilde{m} = 2$ we compare the SER and the score ratio of $G_{\mathrm{COS}(\alpha)}$ and $G_{\mathrm{COS}(\alpha)+}$ depending on α with the ones of G_{OS} and G_{TCOS} in Figure 9. Then in Figure 10, the criteria are represented depending on n (with $\tilde{m} = 2.5$), and on $\tilde{m}$ (with $n = 16$) for every algorithms: here, just three different values of α are considered (0.001, 0.05 and 0.75).

Results 6 *In average, G_{TCOS} has a slightly lower score than G_{OS} (cf. Figure 9 (b)), but it is more accurate (cf. Figure 9 (c), Figures 10 (c) and (d)): it implies that global optima contain in general extra edges. Hence, Algorithm 2 is preferable if a sound constraint is known.*

This important result emphasizes again that the optima of a score function with finite data are not the true networks usually: some false edges improve the score of a graph. Therefore, struc-

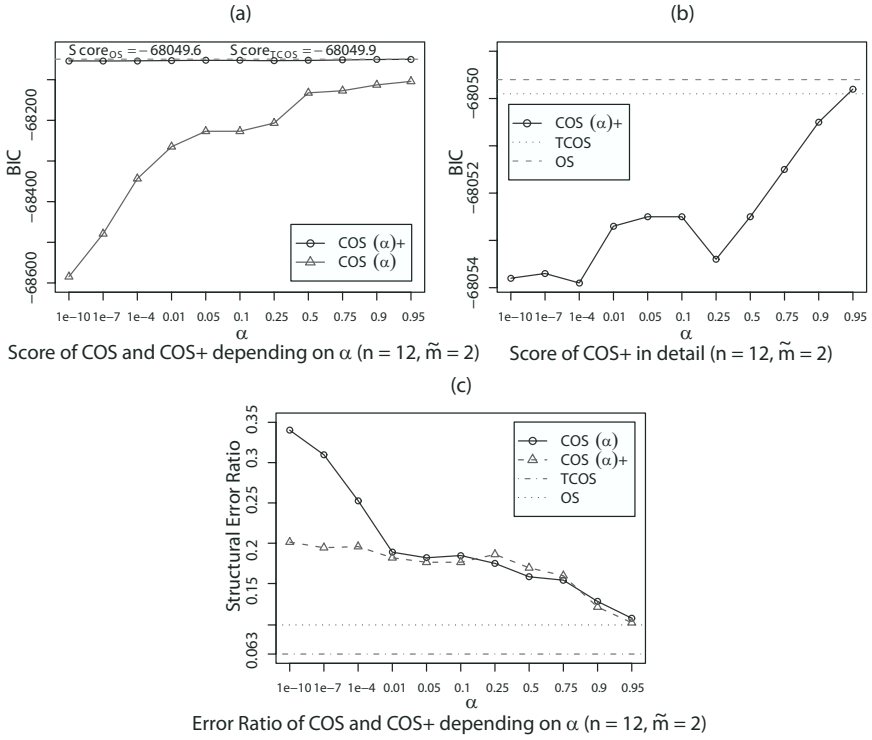


Figure 9: Score and SER for COS(α) and COS(α)+

tural constraint should be generalized whenever a sound super-structure is known: by doing so, the resulting graphs although having a lower score can be more accurate. It is the case for TCOS.

Results 7 *Although* $Score(G_{\mathrm{COS}(\alpha)})$ *converges to* $Score(G_{\mathrm{OS}})$ *when* α *increases (cf. Figure 9(a)), it is usually far lower, and worsen with larger networks or denser networks (cf. Figures 10 (a) and (b)). However,* $SER(G_{\mathrm{COS}(\alpha)})$ *is converging faster to* $SER(G_{\mathrm{OS}})$ *when* α *increases, which enables to find relatively accurate results for* $\alpha \geq 0.01$ *(cf. Figure 9(c)). In addition, n does not affect sensitively* $SER(G_{\mathrm{COS}(\alpha)})$ *(cf. Figure 10 (c)), neither does* $\tilde{m}$ *if* α *is enough high, that is,* $\alpha \geq 0.05$ *(cf. Figure 10 (d)).*

At first sight, these results sound really negative with regard to COS algorithm, or more exactly, with regard to the super-structure learned by MMPC. However, although $Score(G_{\mathrm{COS}(\alpha)})$ is disappointingly low, $SER(G_{\mathrm{COS}(\alpha)})$ is still relatively good: while $SER(G_{\mathrm{OS}}) \approx 0.1$, $SER(G_{\mathrm{COS}}) \approx 0.2$ for α as small as 0.01 in Figure 9 (c). Of course, following Results 6, we could think that for enough high α, $SER(G_{\mathrm{COS}(\alpha)}) \leq SER(G_{\mathrm{OS}})$, but it seems to be hardly feasible while using MMPC: in this case, when IT are relaxed (i.e., $\alpha \to 1$), false edges that improve the score are learned faster than the true ones missing. Still, as it appears in Figures 10 (c) and (d), for $\alpha = 0.75$, COS finds graphs nearly as accurate as the optimal ones. Therefore, when OS is not feasible, COS could be an alternative to greedy searches (such situation is studied in following Sections). Interestingly, our assumption about the potential interest of a hill climbing starting from $G_{\mathrm{COS}(\alpha)}$ are encouraged since $G_{\mathrm{COS}(\alpha)}$ has a low score while being relatively accurate. Next, we focus on COS+.

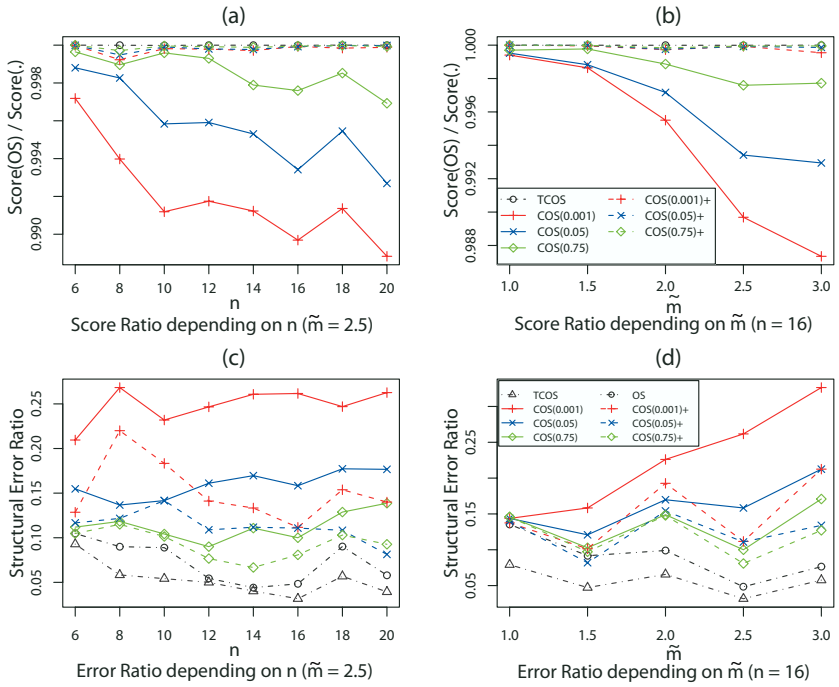


Figure 10: Evolution of Score and SER for every algorithm depending on n and $\tilde{m}$.

Results 8 *For any* α*, after post-processing Score*$(G_{\mathrm{COS}(\alpha)+})$ *is nearly as good as Score*(G_{OS}) *(cf. Figure 9 (a), Figures 10 (a) and (b)).* $G_{\mathrm{COS}(\alpha)+}$ *is in general more accurate than* $G_{\mathrm{COS}(\alpha)}$ *(cf. Figures 10 (c) and (d)), but it is not always the case. Still, SER is clearly improved for* $\alpha < 0.01$ *(cf. Figure 9 (c)): COS+ could be feasible on larger networks while giving interesting results if sufficiently small* α *are used.*

As expected, $G_{\mathrm{COS}(\alpha)}$ is not a local optima, and is at least a good starting point to maximize the score. Figure 9 (b) presents a detailed view of *Score*$(G_{\mathrm{COS}(\alpha)+})$: with post processing, we cannot be sure that *Score*$(G_{\mathrm{COS}(\alpha)+})$ increases strictly with α because of the greedy nature of hill climbing. For the same reasons, previous results concerning the SER are true only in average. To complete Results 8, we should remark that post-processing also appears to improve SER for dense graphs (cf. Figure 10 (d)). This is probably due to the fact that MMPC misses many true edges when the structure is dense: the greedy search would add more true edges missing than false ones, improving consequently the SER.

To summarize this section, TCOS is superior to OS in terms of structure accuracy, while performing faster. However, when S is learned from data, COS cannot perform as well as OS. One efficient strategy to improve both score and accuracy of G_{COS} is to proceed to a post-processing unconstrained hill climbing search. With such an improvement, both score and accuracy of $G_{\mathrm{COS}(\alpha)+}$ become similar for any α used, despite a relative superiority for higher significance levels. Consequently, it would be interesting to compare COS and COS+ to greedy searches, to see if they enable to learn efficiently more accurate graphs than other methods, while being given incomplete super-structures.

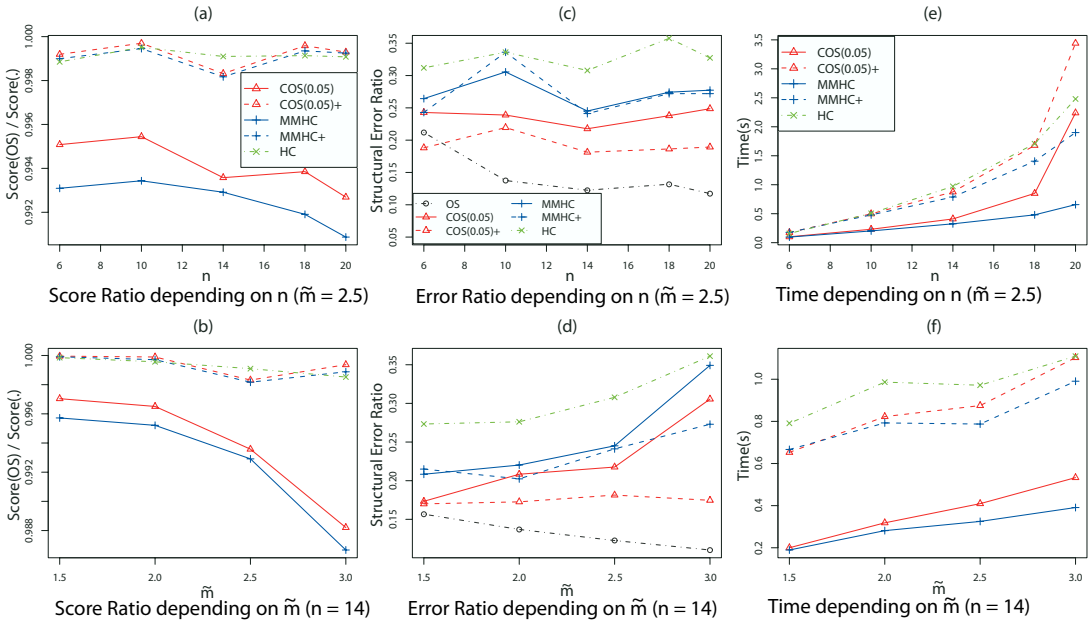


Figure 11: Score, SER and time of every algorithm depending on n and $\tilde{m}$.

5.4 COS and COS+ Compared to Heuristic Searches

In this last series of experiment, we compare COS and COS+ to other greedy searches. The structural constraint is learned with MMPC and $\alpha = 0.05$, it is used by COS and also by a constrained hill climbing search (MMHC). Like COS+, G_{MMHC} is used to start an unconstrained hill climbing (MMHC+). Finally, a classic hill climbing search from the empty graph is also considered (HC). Actually, OS is also performed, but it is not directly referred in this section. Every $n \in [6,10,14,18,20]$, $\tilde{m} \in [1,1.5,2,2.5,3]$, $d \in [500,1000,5000,10000]$ are considered and criteria are averaged over $g = 30$ graphs: in total 3000 graphs are used.

Figure 11 presents the three criteria in function of n and $\tilde{m}$; here $d = 5000$. Synoptic results are presented in Figures 12 and 13, and summarized in Tables 3 and 4. To obtain these tables, we ordered algorithms by score for all the 100 triplets $(n,\tilde{m},d)$ (the scores after averaging over the g graphs were used). Subsequently, we counted the number of times each algorithm was at a given rank: If two algorithms had equal results, they were given the same rank (cf. Table 3). We then performed a similar ranking by comparing SER; the first ranked algorithm being the one that minimized SER (cf. Table 4). In Figures 12 and 13 ranks are represented by colors and all algorithms can be directly compared for every triplet $(n,\tilde{m},d)$. However, while considering the impact of these results, one should take the fact that criteria are compared after averaging into account since it accentuates the contrast between algorithms.

Results 9 *In general,* $Score(G_{\text{COS+}}) \geq Score(G_{\text{MMHC+}}) \geq Score(G_{\text{HC}}) \geq Score(G_{\text{COS}}) \geq Score(G_{\text{MMHC}})$ *(cf. Figures 11 (a), (b), 12 and Table 3). In other words, the super-structure penalizes the score in comparison with HC; however, starting a greedy search from a constrained optimal graph enables to find better scores than HC.*

Many of these inequalities are naturally following the definition of the algorithms; another part comes from the fact that S is incomplete and penalizes the score of the results. Moreover, our

Algorithm	1st	2nd	3rd	4th	5th
COS+	84	8	8	0	0
MMHC+	12	70	18	0	0
HC	7	28	65	0	0
COS	0	1	8	91	0
MMHC	0	0	1	8	91

Table 3: Classification by score.

Algorithm	1st	2nd	3rd	4th	5th
COS+	85	9	6	0	0
COS	14	52	22	12	0
MMHC+	6	29	28	35	2
MMHC	0	12	34	31	23
HC	1	2	17	24	56

Table 4: Classification by SER.

experiments confirm that in nearly every case $Score(G_{\text{COS+}}) > Score(G_{\text{MMHC+}}) > Score(G_{\text{HC}})$; A reason could be that both constrained algorithms find good constrained optima, which are not locally optimal when withdrawing the constraint. Then, since G_{COS} is better than G_{MMHC}, it leads to better local optima when applying the HC post-processing.

Results 10 *In general,* $SER(G_{\text{COS+}}) \leq SER(G_{\text{COS}}) \leq SER(G_{\text{MMHC+}}) \leq SER(G_{\text{MMHC}}) \leq SER(G_{\text{HC}})$ *(cf. Figures 11 (c), (d), 13 and Table 4). The superiority of COS and COS+ is particularly clear with larger data sets (cf. Figure 13).*

Previous results are experimentally more confused than Results 9 since there are no theoretical evidences about SER while comparing algorithms, especially as regards to MMHC+ and MMHC. However, COS+ is clearly demonstrated to be in general the most accurate search. Interestingly, COS+ can be viewed as a bridge between OS (when $\alpha = 1$) and HC (when $\alpha = 0$). Of course, as long as OS is feasible, and without a sound constraint, COS+ is not really needed. However, it is of a certain interest for $n > 30$. Its superiority as compared to HC comes from the fact that some parent sets are optimally learned by COS even with the constraint. Such parent sets could not be learned by the greedy strategy of HC in any cases: as for example in the case of a XOR structure or any configurations for which HC should add several parents in the same time to obtain a score improvement. That way, a HC starting from G_{COS} would benefit from such correct parent sets and find more accurate graphs than a HC starting from the empty graph.

Although for the small networks considered here ($n \leq 20$) our algorithms are as fast as other greedy approaches (cf. Figures 11 (e) and (f)), of course their exponential complexity keeps us from applying them to really large networks without decreasing α. Therefore, COS is restricted to be used only for small networks or sparse ones (Figure 5 (d) gives an idea of what kind of graphs can be considered). COS+ could be used for larger graphs by using lower α as we will see in the next experiment.

5.5 A Real Case: The ALARM Network

In this last experiment, we illustrate a practical usage of COS and COS+ by studying a well-known Bayesian network example, the ALARM network (Beinlich et al., 1989). This graph has $n = 37$ discrete variables having from 2 to 4 states: it is too large to be learned by OS. The maximal in-degree is $c = 4$, the maximal neighborhood is $m = 6$, and the structure is relatively dense since $\tilde{m} \approx 2.5$. Incidentally, the true skeleton entails $86818458 \approx 2^{26}$ connected subsets: if it was given as a prior knowledge, Algorithm 2 could be applied. However, as in previous experiments, S is learned by using MMPC. Nevertheless, α should be enough small so that the super-structure is sparse enough to let execute COS.

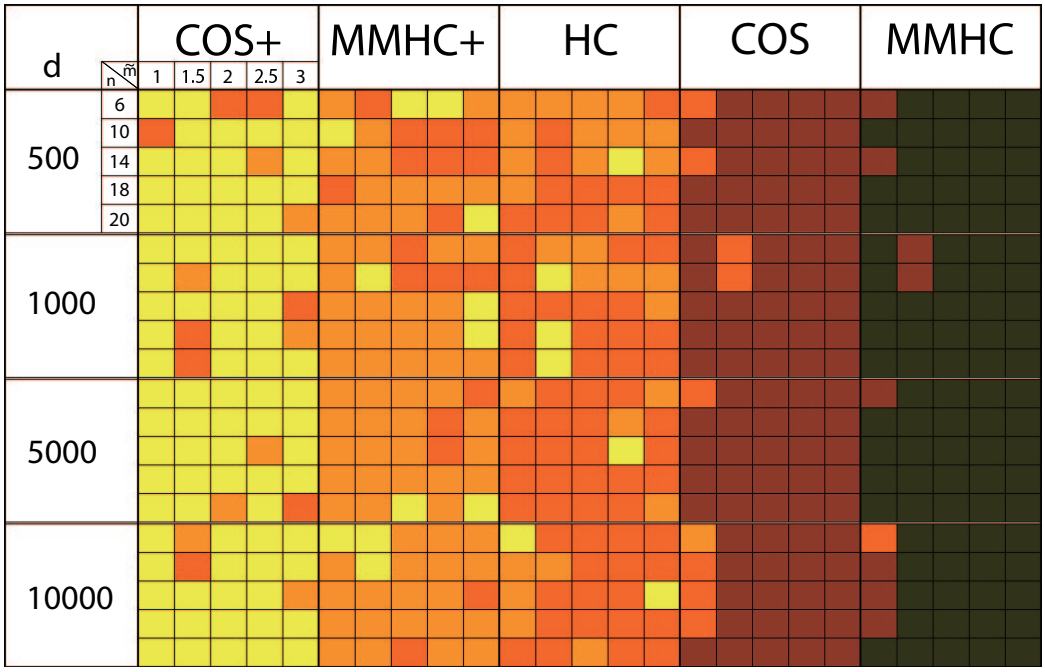


Figure 12: Comparison by score: for every $(n, \tilde{m}, d)$, the worse is the rank of an algorithm (i.e., the lower is the score), the darker is the box.

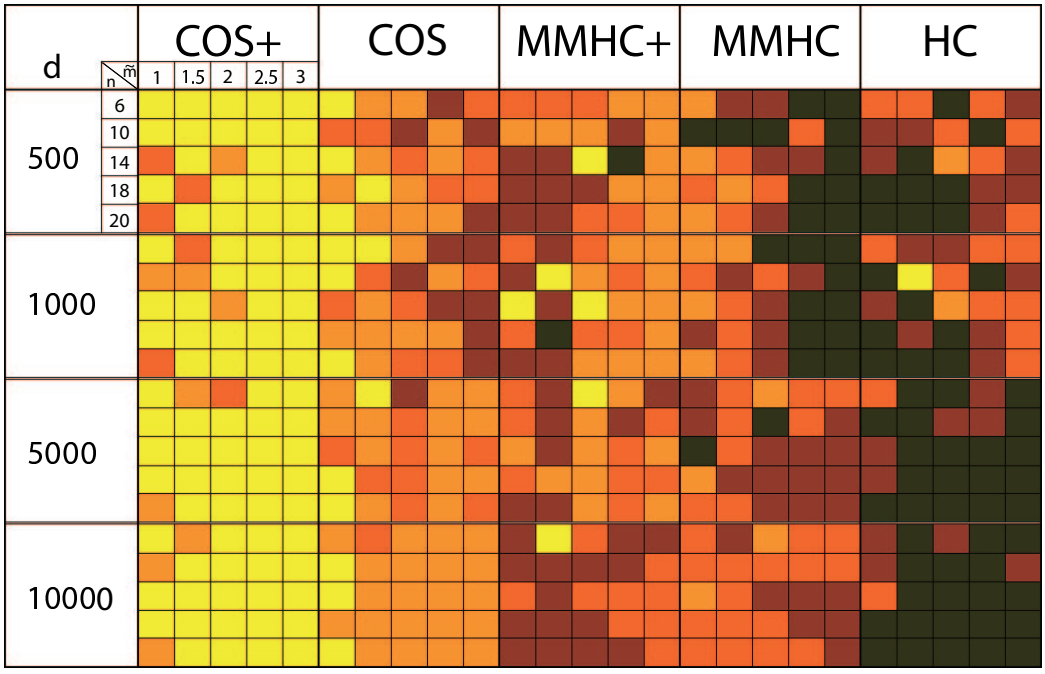


Figure 13: Comparison by SER: for every $(n, \tilde{m}, d)$, the worse is the rank of an algorithm (i.e., the higher is the SER), the darker is the box.

		COS	COS+	MMHC	MMHC+	HC
$d = 3000$	BIC	-33340.5	-32849.5	-33492	-33116.1	-33299.3
$\alpha = 10^{-4}$	SER	0.15	0.12	0.29	0.38	0.45
$t = 24$	Time(s)	4.6	7.1	3.6	6.0	5.0
$d = 3000$	BIC	-33680.8	-32821.4	-33653.2	-33133.2	-33349.9
$\alpha = 10^{-7}$	SER	0.17	0.10	0.29	0.39	0.47
$t = 4$	Time(s)	3.5	6.0	2.7	5.3	5.1
$d = 10000$	BIC	-107834	-106246	-108035	-106808	-107383
$\alpha = 10^{-7}$	SER	0.13	0.08	0.26	0.37	0.51
$t = 270$	Time(s)	10.5	19.2	14.7	23.4	18.8
$d = 10000$	BIC	-108753	-106468	-108420	-107066	-107571
$\alpha = 10^{-10}$	SER	0.15	0.10	0.25	0.35	0.49
$t = 72$	Time(s)	9.1	17.7	12.8	21.4	18.0

Table 5: Averaged criteria for some pairs (d, α) considered.

We consider the same algorithms than in Section 5.4, apart from OS. The significance level used to learn the skeleton of MMHC is always set to 0.01, while various small values of α are tested when learning the super-structure used by COS. For every $d \in [3000, 5000, 10000]$, we start by learning S with MMPC(α) to run COS afterwards. Since, the execution time of COS can be huge depending on S, COS is stopped if its execution exceeds 20 seconds (on a cluster of 96 CPUs of 1050 MHz each, with a total memory of 288GB). In that case, the experience is restarted on a new set of data. If COS finishes on time, all other algorithms are also applied. Previous steps are repeated 20 times for every pair (d, α). We also count the number t of times that COS had to be restarted. Finally, the criteria are averaged over the 20 learned graphs as in the previous Sections.

In all cases results were sensibly the same: Table 5 presents the results obtained with the minimal and maximal values of d and α considered. With larger data sets, smaller significant levels are required in order to obtain an enough sparse S since true edges are learned more confidently. This is why, we can observe that t decreases with smaller α and increases with larger d (cf. Table 5). Therefore, for $d = 10000$ we tried values of $\alpha \in [10^{-7}, \cdots, 10^{-10}]$, while for other data sets $\alpha \in [10^{-4}, \cdots, 10^{-7}]$. In Figures 14 (a) and (b), box plot is used to layout the SER and score of every algorithm over the 20 repetitions of structure learning for $d = 5000$ and $\alpha = 10^{-7}$. Figures 14 (c) and (d) represent the SER of COS and COS+ depending on d and α, respectively.

Results 11 *COS+ is the best algorithm in terms of score, followed by MMHC+ (cf. Table 4 and Figure 14 (a)), while COS+ and COS learn the most accurate networks (cf. Table 4 and Figure 14 (b)). Accuracy of these algorithms is not markedly modified by smaller α (cf. Figure (d)) and slightly improves with bigger data sets (cf. Figure 14 (c)).*

In other words, the present results are in total agreement with the ones derived from smaller and random networks in the previous section. Actually, with ALARM network the contrast between algorithms is even clearer. This is probably because random networks are harder to learn than real ones since the parameters randomly selected can entail nearly unfaithful probability distributions. Interestingly, even with the smallest α used (i.e., $\alpha = 10^{-10}$ in the case of $d = 10000$), only around

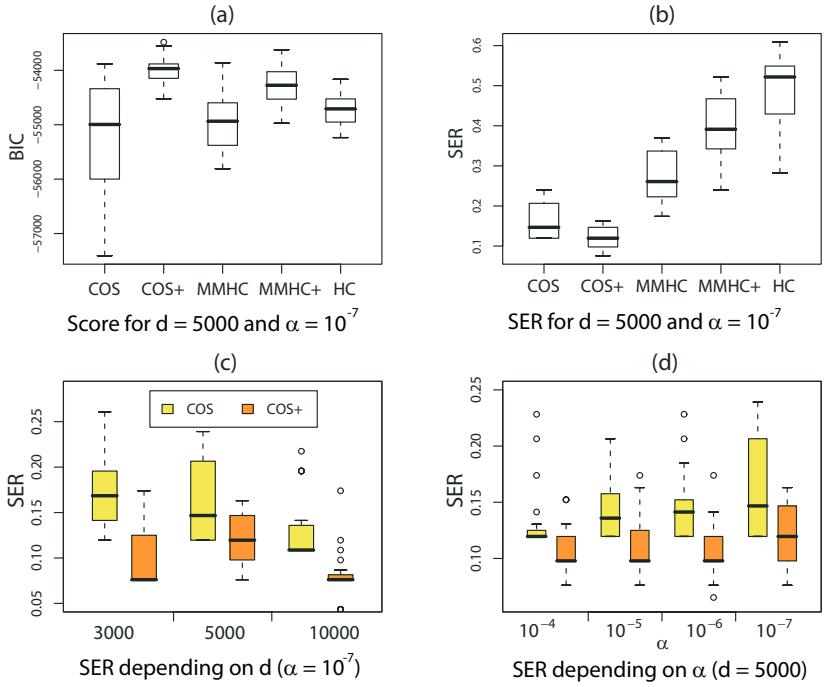


Figure 14: Some detailed results on ALARM network.

6 edges were missing to S in order to be sound. Besides, although COS ran in few seconds for some S, one extra edge in the same super-structure could lead to drastically longer computations. Such a problem is easily understood in terms of the number of connected subsets and underline the fact that some edges contribute particularly to the increase of $|Con(S)|$. Therefore it could be interesting to develop a method to select and withdraw some edges from a learned super-structure, to enable COS+ for larger networks: this way, extremely small α would not be required anymore.

6. Discussion and Conclusion

To conclude our discussion, we would like to recall and summarize the main results of our actual research. First, it is possible to reduce the complexity of an optimal search of an exponential factor by using a structural constraint such as a super-structure. It is then possible to consider larger networks having a sparse skeleton. Moreover, if this super-structure is sound, the accuracy of the resulting graph is improved. Consequently, more attention should be paid to learning sound super-structures rather than true skeleton from data. This should be an easier task and it might improve both speed and accuracy of other search strategies as well, except greedy HC. Next, we outline bellow some strategies that could benefit from a sound super-structure.

In addition, current IT methods that learn a skeleton can be used for approximating a super-structure by relaxing the independency testing. However, as revealed our experiments with MMPC, sound super-structures are rarely learned except for high values of α that implies denser structures and especially a long computation. Although some other IT approaches, such as the randomized version of GS (Margaritis and Thrun, 2000), could solve the problem of complexity, they would probably face the same difficulty to learn efficiently at least every true edge. Indeed, they were de-

signed from the viewpoint of learning the true skeleton, and thereby they should also reject potential extra edges. Learning sound and sparse super-structures is a problem that requires to be considered as a whole, which is not our main concern in the present paper. Therefore, to offset the incompleteness of the super-structures learned with MMPC that weakens the results of our algorithm COS, we developed a greedy post processed search COS+. This algorithm enables to balance between speed and accuracy as it sets up a bridge between optimal search and hill climbing search. In practice, COS+ is demonstrated with success on the ALARM network. It is theoretically feasible for graphs of any sizes but leads to a problem for selecting the significance level; further, we expect that the larger is the graph, the less COS+ should improve over HC.

Therefore, our future research will concentrate on an elaboration of new greedy strategies that benefit from super-structure constraint. As shown in Section 4.2, if S is not having a very high maximal degree (i.e., $m < 20$), F can be calculated even for large networks. Then, by using formula (3) of Section 3.2 we can build in linear time the best constrained graph of a given ordering. This way a fast greedy search over topological orderings is efficiently feasible. With a good set of greedy operations, a constrained optimal graph could be quickly approached using a hill climbing over orderings, without having to calculate M values. Moreover, even for higher m, by fixing a limited number of parents c, we could also manage to calculate F. Consequently, such ordering-based strategy is theoretically feasible for any network size and only need to be evaluated experimentally.

Our second idea concerns only constrained optimal searches. When looking for an optimal graph on $\mathbf{A} \in Con(S)$, if there exists $X_i \in \mathbf{A}$ such that $\mathbf{A}\backslash\{X_i\}$ is unconnected, an optimal graph can be found on another manner. We just need to consider every candidate parent set on the neighborhood of X_i, and search for each of them separately an optimal graph on each connected component of $\mathbf{A}\backslash\{X_i\}$. Thus, if S is a tree, it would be feasible to find an optimal graph in polynomial time with this different strategy. Then, we could develop an algorithm that would change S into a tree, learn an optimal graph that would be post-processed greedily to add missing edges.

Besides, as illustrated in Section 4.4, COS can be used for large graphs, if S is sufficiently sparse. Alternatively, when learning a super-structure, a "score" could be given to every edge that would evaluate the "strength" of the dependency represented. For example, the highest p-value encountered while learning the super-structure with an IT approach could be used. Thus, COS could be used sequentially: the subset of the strongest edges would be considered first, while learning a temporary optimal graph that would be used as a prior knowledge for a second search, this time optimally "adding" a second set of edges. Therefore, by considering successively all the edges of S by a sufficient number of sparse layers, we might be able to approximate accurately an optimal graph. This strategy is potentially interesting since it always assumes the graph as a whole, although the layer of edges should be mainly unconnected to allow COS to be applied to large networks. However, other strategies such as rebuilding optimally the structure locally before merging results could also be developed.

Finally if any of these algorithms were giving convincing empirical proofs of their capacities to learn accurately large networks, we also would like to design strategies for learning sparse and sound super-structures.

Acknowledgments

We would like to thank the reviewers and the associate editor who helped us considerably to improve the quality of this paper, and gave us many valuable suggestions. We should also thank the Super Computer System, Human Genome Center, Institute of Medical Science, University of Tokyo, for letting us use their super computer with which we realized our experiments. We also acknowledge the Japanese Ministry of Education, Culture, Sports, Science and Technology for its financial support.

Appendix A.

Proof of Proposition 1. The complete proof we found is long, complex, and not of the greatest theoretical interest. Hence we will only present the main steps of the demonstration here without explicitly developing every step.

- We start by considering any binary tree, which can be generated by successively applying the following operation to the tree reduced to one leaf: Select a leaf, and change it to a node with two leaves. Then, for the set of trees that have been generated by applying this transformation t times, we define $U(t)$ as the maximum number of sub-trees in these trees, and $V(t)$ as the maximum number of sub-trees that include a given leaf. Given such a tree T, we apply the transformation to a leaf L_i. This generates a node N_i' and two leaves L_{i1}' and L_{i2}', where the prime indicates that we refer to the elements of the newly generated tree T'. By calling $Sub(L)$ the set of sub-trees that contain L, we have: since $|Sub(L_i)| \leq V(t)$, $|Sub(L_{i1}')| \leq 1+2V(t)$. If L_i had a brother L_j, we can show that $|Sub(L_j')| \leq \frac{5}{2}V(t)$ since only half the sub-trees of $Sub(L_j)$ contain L_i. Then, for leaves L_k at agreater distance from L_i than L_j, since the proportion of sub-trees of $Sub(L_k)$ that also contain L_i is decreasing, $Sub(L_k')$ is less increased. Thus, we can conclude that $V(t+1) < \frac{5}{2}V(t)$, and that $V(t) < U(t) < O((\frac{5}{2})^t)$. Finally, since $n = 2t-1$, we derive that $|Con(S)| < O({\alpha_3}^n)$ with $\alpha_3 = \sqrt{\frac{5}{2}}$.
- Hereafter, we can apply the same reasoning to any k-tree, and find that $|Con(S)| < O({\alpha_{k+1}}^n)$ with $\alpha_{k+1} = (\frac{2^k+1}{2})^{\frac{1}{k}}$.
- Then, start the fastidious part of the demonstration by considering a connected forest S of maximal degree m. The idea of the demonstration is to first show that when we consider two nodes of the tree, we can transfer from one node to the other every sub-rooted trees while only increasing the number of connected subsets (actually it is possible to express exactly the variation in the number of connected subsets depending on the unique path between the two nodes and sub-rooted trees of the nodes in this path). Then, by selecting one of the nodes of degree m from S, and fixing it as the root R_0, we can sequentially build a nearly m-1-tree S', except for the root, and maybe for one node R_i, while increasing the number of connected subsets. We do this by considering depth by depth descendants of R_0 and transferring sub-rooted trees from one to the other until all of them at a given depth have $m-1$ or 0 children, taking sub-rooted trees deeper if necessary. Then, if the sons of R_i are not all leaves, we can continue to apply transformations between it and its descendants until obtaining a nearly m-1-tree S" except for its root, and one of its nodes that contain only p leaves. Since $p \leq m-1$, we add $m-p$ leaves to this node, and we select as the root of this final tree S_f one of the

leaves; S_f is like a m-1-tree except for the root that has only one son. In such a case, we can use the results found for such structures, neither the root will sensitively affect the number of connected subsets, nor the nodes that we added at the end; because they can be put in the constant of $O(\alpha_m{}^n)$, which is an upper bound since we built S_f always taking care to increase the number of connected subsets. ■

References

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723, 1974.

I. A. Beinlich, H. Suermondt, R. Chavez, G. Cooper, and et al. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Second European Conference in Artificial Intelligence in Medicine*, 1989.

A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. The travelling salesman problem in bounded degree graphs. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, 2008.

R. Bouckaert. *Bayesian Belief Networks from Construction to Inference*. PhD thesis, University of Utrecht, 1995.

J. Cheng, R. Greiner, J. Kelly, D.A. Bell, and W. Liu. Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

D. Chickering. Learning Bayesian networks is NP-complete. *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130, 1996.

D.M. Chickering. A transformational characterization of equivalent Bayesian network structures. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 87–98. Morgan Kaufman, 1995.

D.M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, pages 445–498, 2002b.

D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. In *Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.

G.F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI-99, 1999.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

C.N. Glymour. *The Mind's Arrows: Bayes Nets & Graphical Causal Models in Psychology*. MIT Press, 2001.

C.N. Glymour and G.F. Cooper. *Computation, Causation, and Discovery*. AAAI Press / The MIT Press, 1999.

D. Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, 1996.

D.E. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

S. Imoto, T. Goto, and S. Miyano. Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. *Pacific Symposium on Biocomputing*, 7:175–186, 2002.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

M. Koivisto and K. Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In S.A. Solla, T.K. Leen, and K.R. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 505–511. MIT Press, 2000.

C. Meek. Strong completeness and faithfulness in Bayesian networks. In *Conference on Uncertainty in Artificial Intelligence*, pages 411–418, 1995.

A. Moore and W. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *Twentieth International Conference on Machine Learning*, ICML-2003, 2003.

R. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

S. Ott and S. Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133, 2003.

S. Ott, S. Imoto, and S. Miyano. Finding optimal models for small gene networks. *Pacific Symposium on Biocomputing*, 9:557–567, 2004.

S. Ott, A. Hansen, S.Y. Kim, and S. Miyano. Superiority of network motifs over optimal networks and an application to the revelation of gene network evolution. *Bioinformatics*, 21(2):227–238, 2005.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman Publishers, San Mateo, CA, 1988.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–671, 1978.

R. Robinson. Counting labelled acyclic digraphs. In *New Directions in the Theory of Graphs*, pages 239–273. Academic Press, 1973.

G. Schwartz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. *Journal of Machine Learning Research*, 6:557–588, 2005.

T. Silander and P. Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Conference on Uncertainty in Artificial Intelligence*, pages 445–452, 2006.

A.P. Singh and A.W. Moore. Finding optimal Bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search*. MIT Press, second edition, 2000.

H. Steck and T. Jaakkola. On the Dirichlet prior and Bayesian regularization. In *Advances in Neural Information Processing Systems*, volume 15, 2002.

J. Suzuki. Learning Bayesian belief networks based on the MDL principle: an efficient algorithm using branch and bound technique. *IEICE Transactions on Information and Systems*, 12, 1998.

M. Teyssier and D. Koller. Ordering-based search: a simple and effective algorithm for learning Bayesian networks. In *Proceedings of the 21th Annual Conference on Uncertainty in Artificial Intelligence*, 2005.

I. Tsamardinos, L.E. Brown, and C.F. Aliferi. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65:31–78, 2006.

Journal of Machine Learning Research 9 (2008) 2287-2320 Submitted 9/07; Published 10/08

Randomized Online PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension*

Manfred K. Warmuth MANFRED@CSE.UCSC.EDU
Dima Kuzmin DIMA@CSE.UCSC.EDU
Computer Science Department
University of California - Santa Cruz
Santa Cruz, CA, 95064

Editor: John Shawe-Taylor

Abstract

We design an online algorithm for Principal Component Analysis. In each trial the current instance is centered and projected into a probabilistically chosen low dimensional subspace. The regret of our online algorithm, that is, the total expected quadratic compression loss of the online algorithm minus the total quadratic compression loss of the batch algorithm, is bounded by a term whose dependence on the dimension of the instances is only logarithmic.

We first develop our methodology in the expert setting of online learning by giving an algorithm for learning as well as the best subset of experts of a certain size. This algorithm is then lifted to the matrix setting where the subsets of experts correspond to subspaces. The algorithm represents the uncertainty over the best subspace as a density matrix whose eigenvalues are bounded. The running time is $O(n^2)$ per trial, where n is the dimension of the instances.

Keywords: principal component analysis, online learning, density matrix, expert setting, quantum Bayes rule

1. Introduction

In Principal Component Analysis (PCA) the n-dimensional data instances are projected into a k-dimensional subspace ($k < n$) so that the total quadratic compression loss is minimized. After centering the data, the problem is equivalent to finding the eigenvectors of the k largest eigenvalues of the data covariance matrix. The variance along an eigendirection is always equal to the corresponding eigenvalue and the subspace defined by the eigenvectors corresponding to the k largest eigenvalues is the subspace that captures the largest total variance and this is equivalent to minimizing the total quadratic compression loss.

We develop a probabilistic online version of PCA: in each trial the algorithm chooses a center m^{t-1} and a k-dimensional projection matrix P^{t-1} based on some internal parameter (which summarizes the information obtained from the previous $t-1$ trials); then an instance x^t is received and the algorithm incurs compression loss $\|(x^t - m^{t-1}) - P^{t-1}(x^t - m^{t-1})\|_2^2$; finally, the internal parameters are updated. The goal is to obtain online algorithms whose total compression loss in all trials is close to the total compression loss $\min_{m,P} \sum_{t=1}^{T} \|(x^t - m) - P(x^t - m)\|_2^2$ of the batch algorithm which can choose its center and k-dimensional subspace in hindsight based on all T instances. Specifically,

*. Supported by NSF grant IIS 0325363. A preliminary version of this paper appeared in Warmuth and Kuzmin (2006b).

in this paper we obtain randomized online algorithms with bounded *regret*. Here we define regret as the difference between the total expected compression loss of the randomized online algorithm and the compression loss of the best mean and subspace of rank k chosen offline. In other words the regret is essentially the expected additional compression loss incurred by the online algorithm compared to normal batch PCA. The expectation is over the internal randomization of the algorithm.

We begin by developing our online PCA algorithm for the uncentered case, that is, all $m^t = 0$ and the compression loss of the offline comparator is simplified to $\min_P \sum_{t=1}^{T} \|x^t - Px^t\|_2^2$. In this simpler case our algorithm is motivated by a related problem in the expert setting of online learning, where our goal is to perform as well as the best size k subset of experts. The algorithm maintains a mixture vector over the n experts. At the beginning of trial t the algorithm chooses a subset P^{t-1} of k experts based on the current mixture vector w^{t-1} that summarizes the previous $t-1$ trials. It then receives a loss vector $\ell^t \in [0,1]^n$. Now the subset P^{t-1} corresponds to the subspace onto which we "compress" or "project" the data. The algorithm incurs no loss on the k components of P^{t-1} and its compression loss equals the sum of the remaining $n-k$ components of the loss vector, that is, $\sum_{i \in \{1,\ldots,n\} - P^{t-1}} \ell_i^t$. Finally it updates its mixture vector to w^t.

The key insight is to maintain a mixture vector w^{t-1} as a parameter with the additional constraint that $w_i^{t-1} \leq \frac{1}{n-k}$. We will show that this "capped" mixture vector represents an implicit mixture over all subsets of experts of size $n-k$, and given w^{t-1} we can efficiently sample a subset of size $n-k$ from the implicit mixture and choose P^{t-1} as the complementary subset of size k. This gives an online algorithm whose total loss over all trials is close to the smallest $n-k$ components of the total loss vector $\sum_{t=1}^{T} \ell^t$. We will show how this algorithm generalizes to an online PCA algorithm when the mixture vector w^{t-1} is replaced by a density matrix W^{t-1} whose eigenvalues are capped by $\frac{1}{n-k}$. Now the constrained density matrix W^{t-1} represents an implicit mixture of $(n-k)$-dimensional subspaces. Again, we can efficiently sample from this mixture, and the complementary k-dimensional subspace P^{t-1} is used for projecting the current instance x_t at trial t.

A simple way to construct an online algorithm is to run the offline or batch algorithm on all data received so far and use the resulting hypothesis on the next data instance. This is called the "Incremental Offline Algorithm" (Azoury and Warmuth, 2001). When the offline algorithm just minimizes the loss on the past instances, then this algorithm is also called the "Follow the Leader (FL) Algorithm" (Kalai and Vempala, 2005). For uncentered PCA we can easily construct a sequence of instances for which the total online compression loss of FL is $\frac{n}{n-k}$ times larger than the total compression loss of batch PCA. However, in this paper we have a more stringent goal. We design randomized online algorithms whose total expected compression loss is at most one times the compression loss of batch PCA plus an additional lower order term which we optimize. In other words, we are seeking online algorithms with bounded *regret*. Our regret bounds are worst-case in that they hold for arbitrary sequences of instances.

Simple online algorithms such as the Generalized Hebbian Algorithm (Sanger, 1989) have been investigated previously that provably converge to the best offline solution. No worst-case regret bounds have been proven for these algorithms. More recently, the online PCA problem was also addressed in Crammer (2006). However, that paper does not fully capture the PCA problem because the presented algorithm uses a full-rank matrix as its hypothesis in each trial, whereas we use a probabilistically chosen projection matrix of the desired rank k. Furthermore, that paper proves bounds on the filtering loss, which are typically easier to obtain, and it is not clear how the filtering loss relates to the more standard regret bounds for the compression loss proven in this paper.

Our algorithm is unique in that we can prove a regret bound for it that is linear in the target dimension k of the subspace but logarithmic in the dimension of the instance space. The key methodology is to use a density matrix as the parameter and employ the quantum relative entropy as a regularizer and measure of progress. This was first done in Tsuda et al. (2005) for a generalization of linear regression to the case when the parameter matrix is a density matrix. Our update of the density matrix can be seen as a "soft" version of computing the top k eigenvectors and eigenvalues of the covariance matrix. It involves matrix logarithms and exponentials which are seemingly more complicated than the FL Algorithm which simply picks the top k directions. Actually, the most expensive step in both algorithms is to update the eigendecomposition of the covariance matrix after each new instance, and this costs $O(n^2)$ time (see, e.g., Gu and Eisenstat, 1994).

The paper is organized as follows. We begin by introducing some basics about batch and online PCA (Section 2) as well as the Hedge Algorithm from the expert setting of online learning (Section 3). We then develop a version of this algorithm that learns as well as the best subset of experts of fixed size (Section 4). When lifted to the matrix setting, this algorithm does uncentered PCA online (Section 5). Surprisingly, the regret bound for the matrix setting stays the same and this is an example of a phenomenon that has been dubbed the "free matrix lunch" (Warmuth, 2007b). We briefly discuss the merits of various alternate algorithms in sections 4.1 and 5.1.

Our online algorithm for centered online PCA is more involved since it has to learn the center as well (Section 6). After motivating the updates to the parameters (Section 6.1) we generalize our regret bound to the centered case (Section 6.2). We then briefly describe how to construct batch PCA algorithms from our online algorithms via standard conversion techniques (Section 6.3). Surprisingly, the bounds obtained this way are competitive with the best known batch PCA bounds. Lower bounds are discussed in Section 7. A brief experimental evaluation is given in Section 8 and we conclude with an overview of online algorithms for matrix parameters and discuss a number of open problems (Section 9).

2. Setup of Batch PCA and Online PCA

Given a set (or batch) of instance vectors $\{x^1, \ldots, x^T\}$, the goal of *batch PCA* is to find a low-dimensional approximation of this data that minimizes the quadratic compression loss. Specifically, we want to find a center vector $m \in \mathbb{R}^n$ and a rank k projection matrix[1] P such that the following loss function is minimized:

$$\text{comp}(P, m) = \sum_{t=1}^{T} \|(x^t - m) - P(x^t - m)\|_2^2. \tag{1}$$

Differentiating and solving for m gives us $m^* = \bar{x}$, where $\bar{x}$ is the data mean. Substituting this optimal center m^* into loss (1) we obtain

$$\begin{aligned} \text{comp}(P) &= \sum_{t=1}^{T} \|(I-P)(x^t - \bar{x})\|_2^2 = \sum_{t=1}^{T} (x^t - \bar{x})^\top (I-P)^2 (x^t - \bar{x}) \\ &= \text{tr}\Big((I-P)^2 \underbrace{\sum_{t=1}^{T} (x^t - \bar{x})(x^t - \bar{x})^\top}_{C} \Big). \end{aligned}$$

1. Projection matrices are symmetric matrices P with eigenvalues in $\{0, 1\}$. Note that $P^2 = P$.

The sum of outer products in the above trace is called the data covariance matrix C. Since $I-P$ is a projection matrix, $(I-P)^2 = I-P$, and

$$\mathrm{comp}(P) = \mathrm{tr}((\underbrace{I-P}_{\text{rank } n-k})C) = \mathrm{tr}(C) - \mathrm{tr}(\underbrace{P}_{\text{rank } k}C).$$

We call the above loss the *compression loss of P* or the *loss of subspace $I-P$*. We now give a justification for this choice of terminology. Observe that $\mathrm{tr}(C)$ equals $\mathrm{tr}(CP)+\mathrm{tr}(C(I-P))$, the sum of the losses of the complementary subspaces. However, we project the data into subspace P and the projected parts of the data are perfectly reconstructed. We charge the subspace P with the parts that are missed, that is, $\mathrm{tr}((I-P)C)$, and therefore call this the compression loss of P.

We now show that $\mathrm{tr}(PC)$ is maximized (or $\mathrm{tr}((I-P)C)$ minimized) if P consists of the k eigendirections of C with the largest eigenvalues. This proof might seem a digression, but elements of it will appear throughout the paper. By rewriting C in terms of its eigendecomposition, that is, $C = \sum_{i=1}^n \gamma_i\, c_i c_i^\top$, we can upper bound $\mathrm{tr}(PC)$ as follows:

$$\mathrm{tr}(PC) = \sum_{i=1}^n \gamma_i\, \mathrm{tr}(P\, c_i c_i^\top) = \sum_{i=1}^n \gamma_i\, c_i^\top P c_i \leq \max_{0\leq\delta_i\leq 1, \sum_i \delta_i = k} \sum_{i=1}^n \gamma_i\, \delta_i.$$

We can replace the scalars $c_i^\top P c_i$ in the ending inequality by the constrained δ_i's because of the following facts:

$$c_i^\top P c_i \leq 1, \text{ for } 1\leq i\leq n, \text{ and } \sum_{i=1}^n c_i^\top P c_i = \mathrm{tr}(P\underbrace{\sum_{i=1}^n c_i c_i^\top}_{I}) = \mathrm{tr}(P) = k,$$

since the eigenvectors c_i of C are an orthogonal set of n directions. A linear function is maximized at one of the vertices of its polytope of feasible solutions. The vertices of this polytope defined by the constraints $0\leq\delta_i\leq 1$ and $\sum_i \delta_i = k$ are those δ vectors with exactly k ones and $n-k$ zeros. Thus the vertices of the polytope correspond to sets of size k and

$$\mathrm{tr}(PC) \;\leq\; \max_{1\leq i_1<i_2<\ldots<i_k\leq n} \sum_{j=1}^k \gamma_{i_j}.$$

Clearly the set that gives the maximum upper bound corresponds to the largest k eigenvalues of C and $\mathrm{tr}(P^*C)$ equals the above upper bound when P^* consists of the eigenvectors corresponding to the set of k largest eigenvalues.

In the online setting, learning proceeds in trials. At trial t the algorithm chooses a center m^{t-1} and a rank k projection matrix P^{t-1}. It then receives an instance x^t and incurs loss

$$\|(x^t - m^{t-1}) - P^{t-1}(x^t - m^{t-1})\|_2^2 = \mathrm{tr}((I-P^{t-1})\,(x^t-m^{t-1})(x^t-m^{t-1})^\top).$$

Note that this is the compression loss of the center m^{t-1} and subspace P^{t-1} on the instance x^t. Our goal is to obtain an algorithm whose total online compression loss over the entire sequence of T trials $\sum_{t=1}^T \mathrm{tr}((I-P^{t-1})\,(x^t-m^{t-1})(x^t-m^{t-1})^\top)$ is close to the total compression loss (1) of the best center m^* and best rank k projection matrix P^* chosen in hindsight by the batch algorithm.

3. Learning as Well as the Best Expert with the Hedge Algorithm

The following setup and algorithm will be the basis of this paper. The algorithm maintains a probability distribution w^{t-1} over n experts. At the beginning of trial t it chooses an expert probabilistically according to the probability vector w^{t-1}, that is, expert i is chosen with probability w_i^{t-1}. Then a loss vector $\ell^t \in [0,1]^n$ is received, where ℓ_i^t specifies the loss of expert i incurred in trial t. The expected loss of the algorithm will be $w^{t-1} \cdot \ell^t$, since the expert was chosen probabilistically. At the end of the trial, the probability distribution is updated to w^t using exponential update factors (See Algorithm 1). This is essentially the Hedge Algorithm of Freund and Schapire (1997). In the

Algorithm 1 Hedge Algorithm

input: Initial n-dimensional probability vector w^0
for $t = 1$ to T **do**
  Draw an expert i with probability w_i^{t-1}
  Receive loss vector ℓ^t
  Incur loss ℓ_i^t
    and expected loss $w^{t-1} \cdot \ell^t$
  $w_i^t = \frac{w_i^{t-1} \exp(-\eta \ell_i^t)}{\sum_{j=1}^n w_j^{t-1} \exp(-\eta \ell_j^t)}$
end for

original version the algorithm proposes a distribution w^{t-1} at trial t and incurs loss $w^{t-1} \cdot \ell^t$ (instead of drawing an expert from w^{t-1} and incurring expected loss $w^{t-1} \cdot \ell^t$).

It is easy to prove the following bound on the total expected loss. Here $d(u,w)$ denotes the relative entropy between two probability vectors $d(u,w) = \sum_{i=1}^n u_i \log \frac{u_i}{w_i}$ and log is the natural logarithm.

Theorem 1 *For an arbitrary sequence of loss vectors $\ell^1, \ldots, \ell^T \in [0,1]^n$, the total expected loss of Algorithm 1 is bounded as follows:*

$$\sum_{t=1}^T w^{t-1} \cdot \ell^t \leq \frac{\eta \sum_{t=1}^T u \cdot \ell^t + \big(d(u,w^0) - d(u,w^T)\big)}{1 - \exp(-\eta)},$$

for any learning rate $\eta > 0$ and comparison vector u in the n dimensional probability simplex.

Proof The update for w^t in Algorithm 1 is essentially the update of the Continuous Weighted Majority Algorithm where the absolute loss of expert i is replaced by ℓ_i^t. Since $\ell_i^t \in [0,1]$, we have $\exp(-\eta \ell_i^t) \leq 1 - (1 - \exp(-\eta))\ell_i^t$ and this implies (essentially Littlestone and Warmuth 1994, Lemma 5.2, or Freund and Schapire 1997):

$$-\log \sum_{i=1}^n w_i^{t-1} \exp(-\eta \ell_i^t) \geq -\log(1 - (1 - \exp(-\eta)) w^{t-1} \cdot \ell^t) \geq w^{t-1} \cdot \ell^t (1 - \exp(-\eta)).$$

The above can be reexpressed with relative entropies as follows (Kivinen and Warmuth, 1999):

$$\begin{aligned} d(u,w^{t-1}) - d(u,w^t) &= -\eta\, u \cdot \ell^t - \log \sum_{i=1}^n w_i^{t-1} \exp(-\eta \ell_i^t) \\ &\geq -\eta\, u \cdot \ell^t + w^{t-1} \cdot \ell^t (1 - \exp(-\eta)). \end{aligned} \tag{2}$$

The bound of theorem can now be obtained by summing over trials. ■

The original Weighted Majority algorithms were described for the absolute loss (Littlestone and Warmuth, 1994). The idea of using loss vectors instead was introduced in Freund and Schapire (1997). The latter paper also shows that when $\sum_t u \cdot \ell^t \leq L$ and $d(u,w^0) - d(u,w^T) \leq D \leq \log n$, then with $\eta = \log(1+\sqrt{2D/L})$, we get the bound

$$\sum_t w^{t-1} \cdot \ell^t \leq \sum_t u \cdot \ell^t + \sqrt{2LD} + d(u,w_0) - d(u,w^T). \tag{3}$$

By setting u to be the vector with a single one identifying the best expert, we get the following bound on the regret of the algorithm (Again log denotes the natural logarithm.):

$$\text{total loss of alg.} - \text{total loss of best expert} \leq \sqrt{2\,(\text{total loss of best expert})\,\log n} + \log n.$$

4. Learning as Well as the Best Subset of Experts

Recall that projection matrices are symmetric positive definite matrices with eigenvalues in $\{0,1\}$. Thus a rank k projection matrix can be written as $P = \sum_{i=1}^{k} p_i p_i^\top$, where the p_i are the k orthonormal vectors forming the basis of the subspace. Assume for the moment that the eigenvectors are restricted to be standard basis vectors. Now a projection matrix becomes a diagonal matrix with k ones in the diagonal and $n-k$ zeros. Also, the trace of a product of such a diagonal projection matrix and any symmetric matrix specifying the loss becomes a dot product between the diagonals of both matrices. The diagonal of the symmetric matrix may be seen as a loss vector ℓ^t. Thus, in this simplified diagonal setting, our goal is to develop online algorithms whose total loss is close to the sum of the lowest $n-k$ components of total loss vector $\sum_{t=1}^{T} \ell^t$. Equivalently, we want to find the highest k components of the total loss vector and per our nomenclature the loss of the lowest $n-k$ components is the compression loss of the complementary highest k components.

For this problem, we will encode the subsets of size $n-k$ as probability vectors: we call $r \in [0,1]^n$ an $(n-k)$-*corner* if it has $n-k$ components fixed to $\frac{1}{n-k}$ and the remaining k components fixed to zero. The algorithm maintains a probability vector w^t as its parameter. At trial t it probabilistically chooses an $(n-k)$-corner r based on the current probability vector w^{t-1} (Details of how this is done will be given shortly). The set of k components missed by r is the set P^{t-1} that we compress with at trial t. The algorithm then receives a loss vector ℓ^t and incurs compression loss $(n-k)\,r \cdot \ell^t = \sum_{i \in \{1,\ldots,n\} - P^{t-1}} \ell_i^t$. Finally the weight vector w^{t-1} is updated to w^t.

We now describe how the corner is chosen: The current probability vector is decomposed into a mixture of n corners and then one of the n corners is chosen probabilistically based on the mixture coefficients. In the description of the decomposition algorithm we use $d = n-k$ for convenience. Let A_d^n denote the convex hull of the $\binom{n}{d}$ corners of size d (where $1 \leq d < n$). Clearly, any component w_i of a vector w in the convex hull is at most $\frac{1}{d}$ because it is a convex combination of numbers in $\{0, \frac{1}{d}\}$. Therefore $A_d^n \subseteq B_d^n$, where B_d^n is the capped probability simplex, that is, the set of n-dimensional vectors w for which $|w| = \sum_i w_i = 1$ and $0 \leq w_i \leq \frac{1}{d}$, for all i. Figure 1 depicts the capped probability simplex for case $d = 2$ and $n = 3, 4$. The following theorem shows that the convex hull of the corners is exactly the capped probability simplex, that is, $A_d^n = B_d^n$. It shows this by expressing any probability vector in the capped simplex B_d^n as a convex combination of at most n $d-$corners. For example, when $d = 2$ and $n = 4$, B_d^n is an octahedron (which has 6 vertices). However, each point in this octahedron is contained in a tetrahedron which is the hull of only 4 of the 6 total vertices.

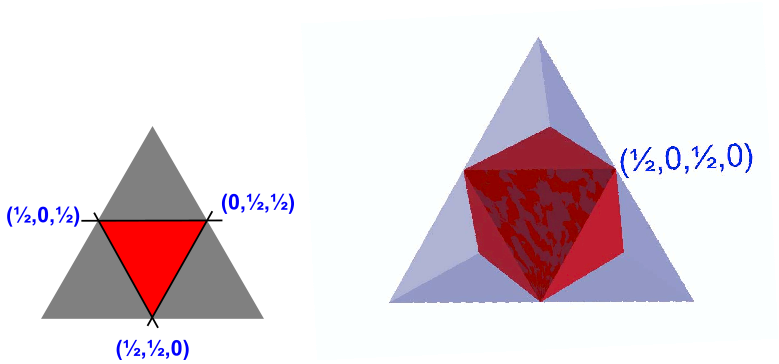


Figure 1: The capped probability simplex B_d^n, for $d = 2$ and $n = 3, 4$. This simplex is the intersection of n halfspaces (one per capped dimension) and its vertices are the $\binom{n}{d}$ d-corners.

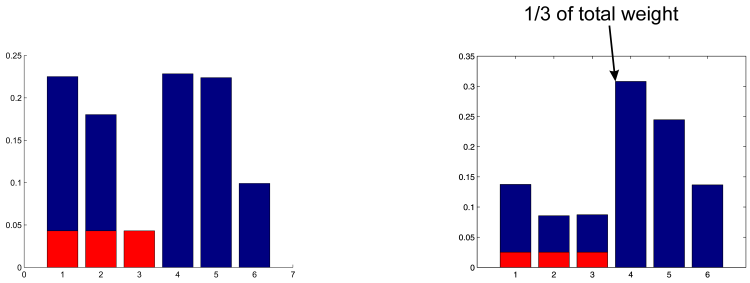


Figure 2: A step of the Mixture Decomposition Algorithm 2, $n = 6$ and $k = 3$. When a corner is removed, then at least one more component is set to zero or raised to a d-th fraction of the total weight. The left picture shows the case where a component inside the corner gets set to zero and the right one depicts the case where a component outside the picked corner gets d-th fraction of the total weight.

Theorem 2 *Algorithm 2 decomposes any probability vector w in the capped probability simplex B_d^n into a convex combination*[2] *of at most n d-corners.*

Proof Let $b(w)$ be the number of *boundary* components in w, that is, $b(w) = |\{i : w_i \text{ is } 0 \text{ or } \frac{|w|}{d}\}|$. Let $\widetilde{B}_d^n$ be all vectors w such that $0 \le w_i \le \frac{|w|}{d}$, for all i. If $b(w) = n$, then w is either a corner or 0. The loop stops when $w = 0$. If w is a corner then it takes one more iteration to arrive at 0. We show that if $w \in \widetilde{B}_d^n$ and w is neither a corner nor 0, then the successor $\widehat{w}$ lies in $\widetilde{B}_d^n$ and $b(\widehat{w}) > b(w)$. Clearly, $\widehat{w} \ge 0$, because the amount that is subtracted in the d components of the corner is at most as large as the corresponding components of w. We next show that $\widehat{w}_i \le \frac{|\widehat{w}|}{d}$. If i belongs to the corner that was chosen then $\widehat{w}_i = w_i - \frac{p}{d} \le \frac{|w|-p}{d} = \frac{|\widehat{w}|}{d}$. Otherwise $\widehat{w}_i = w_i \le l$, and $l \le \frac{|\widehat{w}|}{d}$ follows from the fact that $p \le |w| - d\,l$. This proves that $\widehat{w} \in \widetilde{B}_d^n$.

2. The existence of a convex combination of at most n corners is implied by Carathéodory's theorem (Rockafellar, 1970), but Algorithm 2 gives an effective construction.

Algorithm 2 Mixture Decomposition

input $1 \leq d < n$ and $w \in B_d^n$
repeat
Let r be a corner for a subset of d non-zero components of w
that includes all components of w equal to $\frac{|w|}{d}$
Let s be the smallest of the d chosen components of r
and l be the largest value of the remaining $n-d$ components
$w := w - \underbrace{\min(d\,s, |w| - d\,l)}_{p}\, r$ and **output** $p\,r$
until $w = 0$

For showing that $b(\widehat{w}) > b(w)$ first observe that all boundary components in w remain boundary components in $\widehat{w}$: zeros stay zeros and if $w_i = \frac{|w|}{d}$ then i is included in the corner and $\widehat{w}_i = \frac{|w|-p}{d} = \frac{|\widehat{w}|}{d}$. However, the number of boundary components is increased at least by one because the components corresponding to s and l are both non-boundary components in w and at least one of them becomes a boundary point in $\widehat{w}$: if $p = d\,s$ then the component corresponding to s in w is $s - \frac{p}{d} = 0$ in $\widehat{w}$, and if $p = |w| - d\,l$ then the component corresponding to l in w is $l = \frac{|w|-p}{d} = \frac{|\widehat{w}|}{d}$. It follows that it may take up to n iterations to arrive at a corner which has n boundary components and one more iteration to arrive at 0. Finally note that there is no weight vector $w \in \widetilde{B}_d^n$ s.t. $b(w) = n-1$ and therefore the size of the produced linear combination is at most n. More precisely, the size is at most $n - b(w)$ if $n - b(w) \leq n-2$ and one if w is a corner.

The algorithm produces a linear combination of $(n-k)$-corners, that is, $w = \sum_j p_j r_j$. Since $p_j \geq 0$ and all $|r_j| = 1$, $\sum_j p_j = 1$ and we actually have a convex combination. ■

It is easy to implement the Mixture Decomposition Algorithm in $O(n^2)$ time: simply sort w and spend $O(n)$ per loop.

The batch algorithm for the set problem simply picks the best set in a greedy fashion.

Fact 1 *For any loss vector ℓ, the following corner has the smallest loss of any convex combination of corners in $A_d^n = B_d^n$: Greedily pick the component of minimum loss (d times).*

How can we use the above mixture decomposition and fact to construct an online algorithm? It seems too hard to maintain information about all $\binom{n}{n-k}$ corners of size $n-k$. However, the best corner is also the best convex combination of corners, that is, the best from the set A_{n-k}^n where each member of this set is given by $\binom{n}{n-k}$ coefficients. Luckily, this set of convex combinations equals the capped probability simplex B_{n-k}^n and it takes only n coefficients to specify a member in B_{n-k}^n. Therefore we can maintain a parameter vector in B_{n-k}^n and for any such capped vector w, Algorithm 2 decomposes it into a convex combination of at most n many $(n-k)$-corners. This means that any algorithm producing a hypothesis vector in B_{n-k}^n can be converted to an efficient algorithm that probabilistically chooses an $(n-k)$-corner.

Algorithm 3 spells out the details for this approach. The algorithm chooses a corner probabilistically and $(n-k)\,w^{t-1} \cdot \ell^t$ is the expected loss at trial t. After updating the weight vector w^{t-1} by multiplying with the factors $\exp(-\eta \ell_i^t)$ and renormalizing, the resulting weight vector $\widehat{w}^t$ might lie outside of the capped probability simplex B_{n-k}^n. We then use a Bregman projection with the relative

Algorithm 3 Capped Hedge Algorithm

input: $1 \leq k < n$ and an initial probability vector $w^0 \in B^n_{n-k}$
for $t = 1$ to T **do**
  Decompose w^{t-1} into a convex combination $\sum_j p_j r_j$ of at most n corners r_j
    by applying Algorithm 2 with $d = n-k$
  Draw a corner $r = r_j$ with probability p_j
  Let P^{t-1} be the k components outside of the drawn corner r
  Receive loss vector ℓ^t
  Incur compression loss $(n-k)\, r \cdot \ell^t = \sum_{i \in \{1,\ldots,n\} \setminus P^{t-1}} \ell^t_i$
    and expected compression loss $(n-k)\, w^{t-1} \cdot \ell^t$
  Update: $\widehat{w}^t_i = \dfrac{w^{t-1}_i \exp(-\eta \ell^t_i)}{\sum_{j=1}^n \exp(-\eta \ell^t_j)}$
    $w^t = \mathrm{cap}_{n-k}(\widehat{w}^t)$ where $\mathrm{cap}_{n-k}(.)$ invokes Algorithm 4
end for

Algorithm 4 Capping Algorithm

input probability vector w, set size d
Let $w^{\downarrow}$ index the vector in decreasing order, that is, $w^{\downarrow}_1 = \max(w)$
if $\max(w) \leq \frac{1}{d}$ **then**
  return w
end if
$i = 1$
repeat
  (* Set first i largest components to $\frac{1}{d}$ and normalize the rest to $\frac{d-i}{d}$ *)
  $\widetilde{w} = w$
  $\widetilde{w}^{\downarrow}_j = \frac{1}{d}$, for $j = 1 \ldots i$
  $\widetilde{w}^{\downarrow}_j := \frac{d-i}{d} \frac{\widetilde{w}^{\downarrow}_j}{\sum_{l=i+1}^n \widetilde{w}^{\downarrow}_l}$, for $j = i+1 \ldots n$
  $i := i+1$
until $\max(\widetilde{w}) \leq \frac{1}{d}$
return $\widetilde{w}$

entropy as the divergence to project the intermediate vector $\widehat{w}^t$ back into B^n_{n-k}:

$$w^t = \operatorname*{argmin}_{w \in B^n_{n-k}} d(w, \widehat{w}^t).$$

This projection can be achieved as follows (Herbster and Warmuth, 2001): find the smallest i s.t. capping the largest i components to $\frac{1}{n-k}$ and rescaling the remaining $n-i$ weights to total weight $1 - \frac{i}{n-k}$ makes none of the rescaled weights go above $\frac{1}{n-k}$. The simplest algorithm starts with sorting the weights and then searches for i (see Algorithm 4). However, a linear time algorithm is given in Herbster and Warmuth (2001)[3] that recursively uses the median.

3. The linear time algorithm of Figure 3 of that paper bounds the weights from below. It is easy to adapt this algorithm to the case of bounding the weights from above (as needed here).

When $k = n-1$ and $d = n-k = 1$, B_1^n is the entire probability simplex. In this case the call to Algorithm 2 and the projection onto B_1^n are vacuous and we get the standard Hedge Algorithm (Algorithm 1) as a degenerate case. Note that $(n-k)\sum_{t=1}^T u \cdot \ell^t$ is the total compression loss of comparator vector u. When u is an $(n-k)$-corner, that is, the uniform distribution on a set of size $n-k$, then $(n-k)\sum_{t=1}^T u \cdot \ell^t$ is the total loss of this set.

Theorem 3 *For an arbitrary sequence of loss vectors $\ell^1, \ldots, \ell^T \in [0,1]^n$, the total expected compression loss of Algorithm 3 is bounded as follows:*

$$(n-k)\sum_{t=1}^T w^{t-1} \cdot \ell^t \leq \frac{\eta(n-k)\sum_{t=1}^T u \cdot \ell^t + (n-k)(d(u,w^0) - d(u,w^T))}{1-\exp(-\eta)},$$

for any learning rate $\eta > 0$ and comparison vector $u \in B_{n-k}^n$.

Proof The update for $\widehat{w}^t$ in Algorithm 3 is the same as update for w^t in Algorithm 1. Therefore we can use inequality (2):

$$d(u,w^{t-1}) - d(u,\widehat{w}^t) \geq -\eta\, u \cdot \ell^t + w^{t-1} \cdot \ell^t (1-\exp(-\eta)).$$

Since the relative entropy is a Bregman divergence (Bregman, 1967; Censor and Lent, 1981), the weight vector w^t is a Bregman projection of vector $\widehat{w}^t$ onto the convex set B_{n-k}^n. For such projections the Generalized Pythagorean Theorem holds (see, e.g., Herbster and Warmuth, 2001, for details):

$$d(u,\widehat{w}^t) \geq d(u,w^t) + d(w^t,\widehat{w}^t).$$

Since Bregman divergences are non-negative, we can drop the $d(w^t,\widehat{w}^t)$ term and get the following inequality:

$$d(u,\widehat{w}^t) - d(u,w^t) \geq 0, \text{ for } u \in B_{n-k}^n.$$

Adding this to the previous inequality we get:

$$d(u,w^{t-1}) - d(u,w^t) \geq -\eta\, u \cdot \ell^t + w^{t-1} \cdot \ell^t (1-\exp(-\eta)).$$

By summing over t, multiplying by $n-k$, and dividing by $1-\exp(-\eta)$, the bound follows. ■

It is easy to see that $(n-k)(d(u,w^0) - d(u,w^T)) \leq (n-k)\log\frac{n}{n-k}$ and this is bounded by $k\log\frac{n}{k}$ when $k \leq n/2$. By tuning η as in (3), we get the following regret bound:

$$\begin{aligned}&\text{(expected total compression loss of alg.) - (total compression loss of best } k\text{-subset)}\\ &\overset{k\leq n/2}{\leq} \sqrt{2(\text{total compression loss of best } k\text{-subset})\,k\log\frac{n}{k}} + k\log\frac{n}{k}. \qquad (4)\end{aligned}$$

The last inequality follows from the fact that $(n-k)\log\frac{n}{n-k} \leq k\log\frac{n}{k}$ when $k \leq n/2$. Note that the dependence on k in the last regret bound is essentially linear and dependence on n is logarithmic.

4.1 Alternate Algorithms for Learning as Well as the Best Subset

The question is whether projections onto the capped probability simplex are really needed. We could simply have one expert for each set of $n-k$ components and run Hedge on the $\binom{n}{n-k}$ set experts, where the loss of a set expert is always the sum of the $n-k$ component losses. The set expert $\{i_1,\ldots,i_{n-k}\}$ receives weight proportional to $\exp(-\sum_{j=1}^{n-k}\ell_{i_j}^{<t}) = \prod_{j=1}^{n-k}\exp(-\ell_{i_j}^{<t})$, where $\ell_q^{<t} = \sum_{p=1}^{t}\ell_q^p$. These product weights can be maintained implicitly: keep one weight per component where the ith component receives weight $\exp(-\ell_i^{<t})$, and use dynamic programming for summing the produced weights over the $\binom{n}{n-k}$ sets and for choosing a random set expert based on the product weights. See, for example, Takimoto and Warmuth (2003) for this type of method. While this dynamic programming algorithm can be made reasonably efficient ($O(n^2(n-k))$ per trial), the range of the losses of the set experts is now $[0,\mathbf{n}-\mathbf{k}]$ and this introduces factors of $\mathbf{n}-\mathbf{k}$ into the tuned regret bound:

$$\sqrt{2(\text{total compression loss of best } k\text{-subset})\,(\mathbf{n}-\mathbf{k})\,k\log\frac{n}{k}} + (\mathbf{n}-\mathbf{k})\,k\log\frac{n}{k}. \tag{5}$$

Curiously enough our new capping trick avoids these additional factors in the regret bound by using only the original n experts whose loss is in [0,1]. We do not know whether the improved regret bound (4) (i.e., no additional $n-k$ factors) also holds for the sketched dynamic programming algorithm. However, the following example shows that the two algorithms produce qualitatively different distributions on the sets.

Assume $n=3$ and $k=1$ and the update factors $\exp(-\eta\ell_i^{<t})$ for experts 1, 2 and 3 are proportional to 1, 2, and 4, respectively, which results in the normalized weight vector $(\frac{1}{7},\frac{2}{7},\frac{4}{7})$. Capping the weights at $\frac{1}{n-k}=\frac{1}{2}$ with Algorithm 4 produces the following vector which is then decomposed via Algorithm 2:

$$(\frac{1}{6},\frac{1}{3},\frac{1}{2}) \;= \frac{1}{3}\underbrace{(\frac{1}{2},0,\frac{1}{2})}_{\text{set }\{1,3\}} + \frac{2}{3}\underbrace{(0,\frac{1}{2},\frac{1}{2})}_{\text{set }\{2,3\}}. \tag{6}$$

On the other hand the product weights $\exp(-\eta\ell_i^{<t})*\exp(-\eta\ell_j^{<t})$ of the dynamic programming algorithm for the three sets $\{1,2\}$, $\{1,3\}$ and $\{2,3\}$ of size 2 are $1*2$, $1*4$, and $2*4$, respectively. That is, the dynamic programming algorithm gives (normalized) probability $\frac{1}{7}$, $\frac{2}{7}$ and $\frac{4}{7}$ to the three sets. Notice that Capped Hedge gives expert 3 probability 1 (since it is included in all corners of the decomposition (6)) and the dynamic programming algorithm gives expert 3 probability $\frac{6}{7}$, the total probability it has assigned to the two sets $\{1,3\}$ and $\{2,3\}$ that contain expert 3.

A second alternate is the Follow the Perturbed Leader (FPL) Algorithm (Kalai and Vempala, 2005). This algorithm adds random perturbations to the losses of the individual experts and then selects the set of minimum perturbed loss as its hypothesis. The algorithm is very efficient since it only has to find the set with minimum perturbed loss. However its regret bound has additional factors in addition to the $\mathbf{n}-\mathbf{k}$ factors appearing in the above bound (5) for the dynamic programming algorithm. For the original Randomized Hedge setting with just n experts (Section 3), a distribution of perturbations was found for which FPL simulates the Hedge exactly (Kalai, 2005; Kuzmin and Warmuth, 2005) and therefore the additional factors can be avoided. However we don't know whether there is a distribution of additive perturbations for which FPL simulates Hedge with set experts.

5. Uncentered Online PCA

We create an online PCA algorithm by lifting our new algorithm for sets of experts based on capped weight vector to the matrix case. Now *matrix corners* are density matrices[4] with d eigenvalues equal to $\frac{1}{d}$ and the rest are 0. Such matrix corners are just rank d projection matrices scaled by $\frac{1}{d}$. (Notice that the number of matrix corners is uncountably infinite.) We define the set $\mathcal{A}_d^n$ as the convex hull of all matrix corners. The maximum eigenvalue of a convex combination of symmetric matrices is at most as large as the maximum eigenvalue of any of the individual matrices (see, e.g., Bhatia, 1997, Corollary III.2.2). Therefore each convex combination of corners is a density matrix whose eigenvalues are bounded by $\frac{1}{d}$ and $\mathcal{A}_d^n \subseteq \mathcal{B}_d^n$, where $\mathcal{B}_d^n$ consists of all density matrices whose maximum eigenvalue is at most $\frac{1}{d}$. Assume we have some density matrix $W \in \mathcal{B}_d^n$ with eigendecomposition $\mathcal{W}\,\mathrm{diag}(\omega)\mathcal{W}^\top$. Algorithm 2 can be applied to the vector of eigenvalues ω of this density matrix. The algorithm decomposes ω into at most n diagonal corners r_j: $\omega = \sum_j p_j r_j$. This convex combination can be turned into a convex combination of matrix corners that decomposes the density matrix: $W = \sum_j p_j \mathcal{W}\,\mathrm{diag}(r_j)\mathcal{W}^\top$. It follows that $\mathcal{A}_d^n = \mathcal{B}_d^n$, as in the diagonal case.

As discussed before, losses can always be viewed in two different ways: the loss of the algorithm at trial t is the compression loss of the chosen projection matrix P^{t-1} or the loss of the complementary subspace $I - P^{t-1}$, that is,

$$\|\underbrace{P^{t-1}}_{\text{rank } k} x^t - x^t\|_2^2 = \mathrm{tr}(\underbrace{(I - P^{t-1})}_{\text{rank } n-k} x^t (x^t)^\top).$$

Our online PCA Algorithm 5 has uncertainty about which subspace of rank $n-k$ is best and it represents this uncertainty by a density matrix $W^{t-1} \in \mathcal{A}_{n-k}^n$, that is, a mixture of $(n-k)$-dimensional matrix corners. The algorithm efficiently samples a subspace of rank $n-k$ from this mixture and uses the complementary subspace P^{t-1} of rank k for compression. The expected compression loss of algorithm will be $(n-k)\mathrm{tr}(W^{t-1}xx^\top)$.

The following lemma shows how to pick the best matrix corner. When $S = \sum_{t=1}^T x^t (x^t)^\top$, then this lemma justifies the choice of the batch PCA algorithm.

Theorem 4 *For any symmetric matrix S, $\min_{W \in \mathcal{B}_d^n} \mathrm{tr}(WS)$ attains its minimum at the matrix corner formed by choosing d orthogonal eigenvectors of S of minimum eigenvalue.*

Proof Let $\lambda^\downarrow(W)$ denote the vector of eigenvalues of W in descending order and let $\lambda^\uparrow(S)$ be the same vector of S but in ascending order. Since both matrices are symmetric, $\mathrm{tr}(WS) \geq \lambda^\downarrow(W) \cdot \lambda^\uparrow(S)$ (Marshall and Olkin 1979, Fact H.1.h of Chapter 9, we will sketch a proof below). Since $\lambda^\downarrow(W) \in B_d^n$, the dot product is minimized and the inequality is tight when W is a d-corner (on the n-dimensional probability simplex) corresponding to the d smallest eigenvalues of S. Also the greedy algorithm finds the solution (see Fact 1 of this paper).

For the sake of completeness, we will sketch a proof of the inequality $\mathrm{tr}(WS) \geq \lambda^\downarrow(W) \cdot \lambda^\uparrow(S)$. We begin by rewriting the trace using an eigendecomposition of both matrices:

$$\mathrm{tr}(WS) = \mathrm{tr}(\sum_i \omega_i w_i w_i^\top \sum_j \sigma_j s_j s_j^\top) = \sum_{i,j} \omega_i \sigma_j \underbrace{(w_i \cdot s_j)^2}_{:=M_{i,j}}.$$

4. Density matrix is a symmetric positive definite matrix of trace 1, that is, they are symmetric matrices whose eigenvalues form a probability vector

The matrix M is *doubly stochastic*, that is, its entries are nonnegative and its rows and columns sum to 1. By Birkhoff's Theorem (see, e.g., Bhatia, 1997), such matrices are the convex combinations of permutations matrices (matrices with a single one in each row and column). Therefore the minimum of this linear function occurs at a permutation, and by a swapping argument one can show that the permutation which minimizes the linear function is the one that matches the ith smallest eigenvalue of W with the $(n-i)$th largest eigenvalue of S. ■

We obtain our algorithm for online PCA (Algorithm 5) by lifting Algorithm 3 for set experts to the matrix setting. The exponential factors used in the updates of the expert setting are replaced by the corresponding matrix version which employs the matrix exponential and matrix logarithm (Warmuth and Kuzmin, 2006a).[5] For any symmetric matrix A with eigendecomposition $\sum_{i=1}^{n} \alpha_i a_i a_i^\top$, the matrix exponential $\mathbf{exp}(A)$ is defined as the symmetric matrix $\sum_{i=1}^{n} \exp(\alpha_i) a_i a_i^\top$. Observe that the matrix exponential $\exp(A)$ (and analogously the matrix logarithm $\mathbf{log}(A)$ for symmetric positive definite A) affects only the eigenvalues and not the eigenvectors of A.

The following theorem shows that for the Bregman projection we can keep the eigensystem fixed. Here the quantum relative entropy $\Delta(U,W) = \mathrm{tr}(U(\mathbf{log}\,U - \mathbf{log}\,W))$ is used as the Bregman divergence.

Theorem 5 *Projecting a density matrix onto $\mathcal{B}_d^n$ w.r.t. the quantum relative entropy is equivalent to projecting the vector of eigenvalues w.r.t. the "normal" relative entropy: If W has the eigendecomposition $\mathcal{W}\,\mathrm{diag}(\omega)\,\mathcal{W}^\top$, then*

$$\operatorname*{argmin}_{U \in \mathcal{B}_d^n} \Delta(U,W) \;=\; \mathcal{W} u^* \mathcal{W}^\top, \text{ where } u^* = \operatorname*{argmin}_{u \in B_d^n} d(u,\omega).$$

Proof The quantum relative entropy can be rewritten as follows:

$$\Delta(U,W) = \mathrm{tr}(U\,\mathbf{log}\,U) - \mathrm{tr}(U\,\mathbf{log}\,W) = \lambda(U)\cdot\log(\lambda(U)) - \mathrm{tr}(U\,\mathbf{log}\,W),$$

where $\lambda(U)$ denotes the vector of eigenvalues of U and log is the componentwise logarithm of a vector. For any symmetric matrices S and T, $\mathrm{tr}(ST) \le \lambda^{\downarrow}(S)\cdot\lambda^{\downarrow}(T)$ (Marshall and Olkin 1979, Fact H.1.g of Chapter 9; also see proof sketch of a similar fact given in previous theorem). This implies that

$$\Delta(U,W) \ge \lambda(U)\cdot\log(\lambda(U)) - \lambda^{\downarrow}(U)\cdot\lambda^{\downarrow}(\mathbf{log}(W)) = \lambda(U)\cdot\log(\lambda(U)) - \lambda^{\downarrow}(U)\cdot\log\lambda^{\downarrow}(W).$$

Therefore $\min_{U \in \mathcal{B}_d^n} \Delta(U,W) \ge \min_{u \in B_d^n} d(u,\omega)$, and if u^* minimizes the r.h.s. then $\mathcal{W}\,\mathrm{diag}(u^*)\,\mathcal{W}^\top$ minimizes the l.h.s. because $\Delta(\mathcal{W}\,\mathrm{diag}(u^*)\,\mathcal{W}, W) = d(u^*,\omega)$. ■

The lemma means that the projection of a density matrix onto $\mathcal{B}_{n-k}^n$ is achieved by applying Algorithm 4 to the vector of eigenvalues of the density matrix.

We are now ready to prove a worst-case loss bound for Algorithm 5 for the uncentered case of online PCA. Note that the expected loss in trial t of this algorithm is $(n-k)\mathrm{tr}(W^{t-1}x^t(x^t)^\top)$. When U is a matrix corner then $(n-k)\sum_{t=1}^{T}\mathrm{tr}(Ux^t(x^t)^\top)$ is the total loss of the corresponding subspace.

5. This update step is a special case of the Matrix Exponentiated Gradient update for the the linear loss $\mathrm{tr}(Wx^t(x^t)^\top)$ (Tsuda et al., 2005).

Algorithm 5 Uncentered online PCA algorithm

input: $1 \leq k < n$ and an initial density matrix $W^0 \in \mathcal{B}^n_{n-k}$
for $t = 1$ to T **do**
  Perform eigendecomposition $W^{t-1} = \mathcal{W}\omega\mathcal{W}^\top$
  Decompose ω into a convex combination $\sum_j p_j r_j$ of at most n corners r_j
    by applying Algorithm 2 with $d = n-k$
  Draw a corner $r = r_j$ with probability p_j
  Form a matrix corner $R = \mathcal{W}\,\mathrm{diag}(r)\,\mathcal{W}^\top$
  Form a rank k projection matrix $P^{t-1} = I - (n-k)R$
  Receive data instance vector x^t
  Incur compression loss $\|x^t - P^{t-1}x^t\|_2^2 = \mathrm{tr}((I - P^{t-1})\,x^t(x^t)^\top)$
    and expected compression loss $(n-k)\mathrm{tr}(W^{t-1}x^t(x^t)^\top)$
  Update: $\widehat{W}^t = \dfrac{\mathbf{exp}(\mathbf{log}\,W^{t-1} - \eta\,x^t(x^t)^\top)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^{t-1} - \eta\,x^t(x^t)^\top))}$
    $W^t = \mathrm{cap}_{n-k}(\widehat{W}^t)$,
      where $\mathrm{cap}_{n-k}(A)$ applies Algorithm 4 to the vector of eigenvalues of A
end for

Theorem 6 *For an arbitrary sequence of data instances $x^1, \ldots, x^T$ of 2-norm at most one, the total expected compression loss of the algorithm is bounded as follows:*

$$\sum_{t=1}^{T}(n-k)\mathrm{tr}(W^{t-1}x^t(x^t)^\top) \leq \frac{\eta(n-k)\sum_{t=1}^{T}\mathrm{tr}(Ux^t(x^t)^\top) + (n-k)(\Delta(U,W^0) - \Delta(U,W^T))}{1-\exp(-\eta)},$$

for any learning rate $\eta > 0$ and comparator density matrix $U \in \mathcal{B}^n_{n-k}$.

Proof The update for $\widehat{W}^t$ is a density matrix version of the Hedge update which was used for variance minimization along a single direction (i.e., $k = n-1$) in Warmuth and Kuzmin (2006a). The basic inequality (2) for that update becomes:

$$\Delta(U,W^{t-1}) - \Delta(U,\widehat{W}^t) \geq -\eta\,\mathrm{tr}(Ux^t(x^t)^\top) + \mathrm{tr}(W^{t-1}x^t(x^t)^\top)(1-\exp(-\eta)).$$

As in the proof of Theorem 3 of this paper, the Generalized Pythagorean Theorem applies and dropping one term we get the following inequality:

$$\Delta(U,\widehat{W}^t) - \Delta(U,W^t) \geq 0, \text{ for } U \in \mathcal{B}^n_{n-k}.$$

Adding this to the previous inequality we get:

$$\Delta(U,W^{t-1}) - \Delta(U,W^t) \geq -\eta\,\mathrm{tr}(Ux^t(x^t)^\top) + \mathrm{tr}(W^{t-1}x^t(x^t)^\top)(1-\exp(-\eta)).$$

By summing over t, multiplying by $n-k$, and dividing by $1-\exp(-\eta)$, the bound follows. ■

It is easy to see that $(n-k)(\Delta(U,W^0)-\Delta(U,W^T)) \leq (n-k)\log\frac{n}{n-k}$ and this is bounded by $k\log\frac{n}{k}$ when $k \leq n/2$. By tuning η as in (3), we can get regret bounds of the form:

$$\begin{aligned}&\text{(expected total compression loss of alg.) - (total compression loss of best } k\text{-subspace)}\\ &\overset{k\leq n/2}{\leq} \sqrt{2(\text{total compression loss of best } k\text{-subspace})\,k\log\frac{n}{k}} + k\log\frac{n}{k}. \end{aligned} \tag{7}$$

Let us complete this section by discussing the minimal assumptions on the loss functions needed for proving the regret bounds obtained so far. Recall that in the regret bounds for experts as well as set experts we always assumed that the loss vector ℓ^t received at trial t lies in $[0,1]^n$. In the case of uncentered PCA, the loss at trial t is specified by an instance vector x^t that has 2-norm at most one. In other words, the single eigenvalue of the instance matrix $x^t(x^t)^\top$ must be bounded by 1. However, it is easy to see that the regret bound of the previous theorem still holds if at trial t the instance matrix $x^t(x^t)^\top$ is replaced by any symmetric instance matrix S^t whose vector of eigenvalues lies in $[0,1]^n$.

5.1 Alternate Algorithms for Uncentered Online PCA

We conjecture that the following algorithm has the regret bound (7) as well: run the dynamic programming algorithm for the set experts sketched in Section 4 on the vector of eigenvalues of the current covariance matrix. The produced set for size k is converted to a projection matrix of rank k by replacing it with the k outer products of the corresponding eigenvectors. We are not elaborating on this approach since the algorithm inherits the additional $n-k$ factors contained in the regret bound (5) for set experts. If these factors in the regret bound for set experts can be eliminated then this approach might lead to a competitive algorithm.

Versions of FPL might also be used to design an online PCA algorithm for compressing with a k dimensional subspace. Such an algorithm would be particularly useful if the same regret bound (7) could be proven for it as for our online PCA algorithm. The question is whether there exists a distribution of additive perturbations of the covariance matrix for which the loss of the subspace formed by the eigenvectors of the $n-k$ smallest eigenvalues simulates a matrix version of Hedge on subspaces of rank $n-k$ and whether this algorithm does not have the $n-k$ factors in its bound. Note that extracting the subspace formed by the eigenvectors of the $n-k$ smallest (or k largest) eigenvalues might be more efficient than performing a full eigendecomposition.

6. Centered Online PCA

In this section we extend our online PCA algorithm to also estimate the data center online. Under the extended protocol, the algorithm needs to produce both a rank k projection matrix P^{t-1} and a data center m^{t-1} at trial t. It then receives a data point x^t and incurs compression loss $\|(x^t-m^{t-1})-P^{t-1}(x^t-m^{t-1})\|_2^2$. As for uncentered online PCA, we will use a capped density matrix W^{t-1} to represent the algorithm's uncertainty about the hidden subspace.

6.1 Motivation of the Updates

We begin by motivating the updates of all the algorithms analyzed so far. We follow Kivinen and Warmuth (1997) and motivate the updates by minimizing a tradeoff between a parameter divergence and a loss function. Here we also have the linear capping constraints. Since our loss is linear, the

tradeoff minimization problem can be solved exactly instead of using approximations as is done in Kivinen and Warmuth (1997) for non-linear losses. Updates motivated by exact solution of tradeoff minimization problems involving non-linear loss functions are sometimes called *implicit* updates since they typically do not have a closed form (Kivinen et al., 2005). Even though the loss function used here is linear, the additional capping constraints are responsible for the fact that there is again no closed form for the updates. Nevertheless our algorithms are always able to compute the optimal solutions of the tradeoff minimization problems defining the updates.

We begin our discussion of motivations of updates with the set expert case. Consider the following two updates:

$$w^t = \text{arginf}_{w \in B^n_{n-k}} \left(\eta^{-1} d(w, w^{t-1}) + w \cdot \ell^t \right), \tag{8}$$

$$w^t = \text{arginf}_{w \in B^n_{n-k}} \left(\eta^{-1} d(w, w^0) + \sum_{q=1}^{t} w \cdot \ell^q \right). \tag{9}$$

In the motivations of all our updates, the divergences are always versions of relative entropies which are special cases of Bregman divergences. Here d denotes the standard relative entropy between probability vectors. The first update above trades off the divergence to the last parameter vector with the loss in the last trial. The second update trades off the divergence to the initial parameter with the total loss in all past trials. In both cases the minimization is over B^n_{n-k} which as we recall is the n-dimensional probability simplex with the components capped at $\frac{1}{n-k}$. One can show that the combined two update steps of the Capped Hedge Algorithm 3 coincide with the first update (8) above. The solution to (8) has the following exponential form:

$$w^t_i = \frac{w^{t-1}_i \exp(-\eta \ell^t_i + \gamma^t_i)}{\sum_{j=1}^n w^{t-1}_j \exp(-\eta \ell^t_j + \gamma^t_j)},$$

where γ^t_i is the Lagrangian coefficient that enforces the cap on the weight w^t_i. The non-negativity constraints don't have to be explicitly enforced because the relative entropy is undefined on vectors with negative elements and thus acts as a barrier function. Because of the capping constraints, the two updates (8) and (9) given above are typically not the same. However when $k = n-1$, then $B^n_{n-k} = B^n_1$ is the entire probability simplex and the γ^t_i coefficients disappear. In that case both updates agree and motivate the update of vanilla Hedge (Algorithm 1) (See Kivinen and Warmuth, 1999).

Furthermore, the above update (8) can be split into two steps as is done in Algorithm 3: the first update step uses exponential factors to update the probability vector and the second step performs a relative entropy projection of the intermediate vector onto the capped probability simplex. Here we give the sequence of two optimization problems that motivate the two update steps of Algorithm 3:

$$\widehat{w}^t = \text{arginf}_{w_i \geq 0,\, \sum w_i = 1} \left(\eta^{-1} d(w, w^{t-1}) + w \cdot \ell^t \right),$$

$$w_t = \text{arginf}_{w \in B^n_{n-k}} d(w, \widehat{w}^t).$$

For the motivation of the uncentered online PCA update (Algorithm 5), we replace the relative entropy $d(w, w^{t-1})$ between probability vectors in (8) by the Quantum Relative Entropy $\Delta(W, W^{t-1}) = \text{tr}(W(\mathbf{log}\, W - \mathbf{log}\, W^{t-1}))$ between density matrices. Furthermore, we change the loss function from a dot product to a trace:

$$W^t = \text{arginf}_{W \in \mathcal{D}^n_{n-k}} \left(\eta^{-1} \Delta(W, W^{t-1}) + \text{tr}(W x^t (x^t)^\top) \right).$$

Recall that $\mathcal{B}^n_{n-k}$ is the set of all $n \times n$ density matrices whose maximum eigenvalue is at most $\frac{1}{n-k}$. Note that in Algorithm 5, this update is again split into two steps.

The case of centered online PCA, which we will address now, is the most interesting because now we have two parameters. We use the following update which uses a divergence to the initial parameters (as in (9)):

$$(W^t, m^t) \;=\; \operatorname*{arginf}_{W \in \mathcal{B}^n_{n-k},\, m \in \mathbb{R}^n} \quad \Big(\eta^{-1}\Delta(W, W^0) + \widetilde{\eta}^{-1}(m-m^0)^\top W (m-m^0)\Big) + \sum_{q=1}^{t} \operatorname{tr}(W(x^q-m)(x^q-m)^\top). \tag{10}$$

Notice that we have two learning rates: η for the density matrix parameter and $\widetilde{\eta}$ for the center parameter. The above update may be viewed as a maximum a posteriori estimator since the divergences act as priors or initial examples and the inverse learning rates that multiply the divergences determine the importance of the priors (See, e.g., Azoury and Warmuth, 2001, for a discussion). When $\eta^{-1} = \widetilde{\eta}^{-1} = 0$, then there are no priors and the update become the Maximum Likelihood estimator or Follow the Leader (FL) Algorithm. If $\widetilde{\eta}^{-1} \to \infty$, then m^t is clamped to the fixed center m^0. If further $m^0 = 0$, then the above motivation becomes a motivation for an uncentered update with a divergence to the initial density matrix W^0 (analogous to (9)). Similarly, when $\eta^{-1} \to \infty$, then W^t is clamped to the fixed density matrix W^0 and the resulting optimization problem motivates the Incremental Off-line Algorithm for Gaussian density estimation with a fixed covariance matrix (Azoury and Warmuth, 2001).

As in Kuzmin and Warmuth (2007), we analyze this update for centered PCA by rewriting its optimization problem (10) as the dual maximization problem. The constraint $W \in \mathcal{B}^n_{n-k}$ in equation (10) is equivalent to having constraints $\operatorname{tr}(W) = 1$ and $W \preceq \frac{1}{n-k} I$. The constraint $W \succeq 0$ is automatically enforced since the quantum relative entropy acts as a barrier. With this in mind, we write down the Lagrangian function, where $U^t(W, m)$ is the objective function of our optimization problem (10) that includes data points from t trials, δ is the dual variable for the trace constraint and the symmetric positive definite matrix Γ is the dual variable for the capping constraint:

$$L^t(W, m, \Gamma, \delta) \;=\; U^t(W, m) + \delta(\operatorname{tr}(W) - 1) + \operatorname{tr}\Big((W - \frac{1}{n-k} I)\Gamma\Big).$$

The optimization over m is unconstrained, giving the solution for m^t:

$$m^t = \frac{\widetilde{\eta}^{-1} m^0 + \sum_{q=1}^t x^q}{\widetilde{\eta}^{-1} + t}. \tag{11}$$

This is essentially the normal mean of an extended sample, where we added $\widetilde{\eta}^{-1}$ copies of m^0 to $x^1, \ldots, x^t$. To write down the form of the solution for W^t compactly we will introduce the following matrix:

$$C^t = \widetilde{\eta}^{-1}(m^0 - m^t)(m^0 - m^t)^\top + \sum_{q=1}^{t} (x^q - m^t)(x^q - m^t)^\top. \tag{12}$$

This can be seen as the extended sample covariance matrix where we added $\widetilde{\eta}^{-1}$ copies of instance m^0.

Setting the derivatives to zero and solving (see Tsuda et al. 2005 for similar derivation), we obtain the following form of W^t in terms of the dual variables $\delta' = \eta\delta$ and Γ:

$$W^t(\delta', \Gamma) = \mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \delta' I - \eta\Gamma).$$

The constraint $\mathrm{tr}(W) = 1$ is enforced by choosing $\delta' = \log \mathrm{tr}(\mathbf{exp}(\mathbf{log} W_1 - \eta C^t - \Gamma))$. By substituting $W^t(\delta', \Gamma)$ and the formula for m^t into the Lagrangian L^t and simplifying, we obtain the following dual problem:

$$\max_{\Gamma \succeq 0} \widehat{L}^t(\Gamma), \text{ where } \widehat{L}^t(\Gamma) = -\eta^{-1} \log \mathrm{tr}(\mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \eta\Gamma)) - \frac{\mathrm{tr}(\Gamma)}{n-k}. \tag{13}$$

Let Γ^t be the optimal solution of the dual problem above and let $\mathrm{cap}_d(W)$ be the density matrix obtained when the capping Algorithm 4 is applied to the vector of eigenvalues of W and capping parameter d. This lets us express W^t as:

$$W^t = \frac{\mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \eta\Gamma^t)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log} W^0 - \eta C^t - \eta\Gamma^t))} = \mathrm{cap}_{n-k}\Big(\frac{\mathbf{exp}(\mathbf{log} W^0 - \eta C^t)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log} W^0 - \eta C^t))}\Big). \tag{14}$$

For the analysis we express m^t and C^t as online updates:

Lemma 7 *The estimates of mean and covariance can be updated as follows:*

$$m^t = \frac{(\widetilde{\eta}^{-1} + t - 1)m^{t-1} + x^t}{\widetilde{\eta}^{-1} + t} = m^{t-1} - \frac{1}{\widetilde{\eta}^{-1} + t}(m^{t-1} - x^t),$$

$$C^t = C^{t-1} + \frac{\widetilde{\eta}^{-1} + t - 1}{\widetilde{\eta}^{-1} + t}(x^t - m^{t-1})(x^t - m^{t-1})^\top$$

Proof The update rule for m^t is easy to verify. For the update of C^t, we start by expanding the expression (12) for C^{t-1}:

$$\begin{aligned} C^{t-1} &= \widetilde{\eta}^{-1}(m^0(m^0)^\top - m^0(m^{t-1})^\top - m^{t-1}(m^0)^\top + m^{t-1}(m^{t-1})^\top) \\ &\quad + \sum_{q=1}^{t-1}(m^{t-1}(m^{t-1})^\top - x^q(m^{t-1})^\top - m^{t-1}(x^q)^\top + x^q(x^q)^\top) \\ &= \sum_{q=1}^{t-1} x^q(x^q)^\top + (\widetilde{\eta}^{-1} + t - 1)m^{t-1}(m^{t-1})^\top \\ &\quad - (\widetilde{\eta}^{-1}m^0 + \sum_{q=1}^{t-1} x^q)(m^{t-1})^\top - m^{t-1}(\widetilde{\eta}^{-1}m^0 + \sum_{q=1}^{t-1} x^q)^\top + \widetilde{\eta}^{-1}m^0(m^0)^\top. \end{aligned}$$

By substituting

$$\widetilde{\eta}^{-1}m^0 + \sum_{q=1}^{t-1} x^q = (\widetilde{\eta}^{-1} + t - 1)m^{t-1}$$

we get the following:

$$C^{t-1} = \sum_{q=1}^{t-1} x^q(x^q)^\top - (\widetilde{\eta}^{-1} + t - 1)m^{t-1}(m^{t-1})^\top + \widetilde{\eta}^{-1}m^0(m^0)^\top.$$

Algorithm 6 Centered Online PCA Algorithm

input: $1 \leq k < n$ and an initial offset m^0, initial density matrix $W^0 \in \mathcal{B}^n_{n-k}$, $C^0 = 0$
for $t = 1$ to T **do**
Perform eigendecomposition $W^{t-1} = \mathcal{W}\omega\mathcal{W}^\top$
Decompose ω into a convex combination $\sum_j p_j r_j$ of at most n corners r_j
by applying Algorithm 2 with $d = n-k$
Draw corner $r = r_j$ with probability p_j
Form a matrix corner $R = \mathcal{W}\,\mathrm{diag}(r)\,\mathcal{W}^\top$
Form a rank k projection matrix $P^{t-1} = I - (n-k)R$
Receive data instance vector x^t
Incur compression loss
$\|(x^t - m^{t-1}) - P^{t-1}(x^t - m^{t-1})\|_2^2 = \mathrm{tr}((I - P^{t-1})(x^t - m^{t-1})(x^t - m^{t-1})^\top)$
and expected compression loss $(n-k)\mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top)$
Update:

$$m^t = m^{t-1} - \frac{1}{\widetilde{\eta}^{-1} + t}(m^{t-1} - x^t) \tag{15}$$

$$C^t = C^{t-1} + \frac{\widetilde{\eta}^{-1} + t - 1}{\widetilde{\eta}^{-1} + t}(x^t - m^{t-1})(x^t - m^{t-1})^\top \tag{16}$$

$$\widehat{W}^t = \frac{\mathbf{exp}(\mathbf{log}\, W^0 - \eta C^t)}{\mathrm{tr}(\mathbf{exp}(\mathbf{log}\, W^0 - \eta C^t))}$$

$$W^t = \mathrm{cap}_{n-k}(\widehat{W}^t),$$

where $\mathrm{cap}_{n-k}(A)$ applies Algorithm 4 to the vector of eigenvalues of A
end for

Now the update for C can be written as:

$$C^t = C^{t-1} + (\widetilde{\eta}^{-1} + t - 1)m^{t-1}(m^{t-1})^\top + x^t(x^t)^\top - (\widetilde{\eta}^{-1} + t)m^t(m^t)^\top.$$

Substituting the left update for m^t from the statement of the lemma and simplifying gives the desired online update for C^t. ■

: All the steps for the Centered Online PCA Algorithm are summarized as Algorithm 6. We already reasoned that the capping and decomposition steps are $O(n^2)$. The remaining expensive step is maintaining the eigendecomposition of the covariance matrix for computing the matrix exponential. Using standard rank one update techniques for the eigendecomposition of a symmetric matrix, this costs $O(n^2)$ per trial (see, e.g., Gu and Eisenstat, 1994).

6.2 Regret Bound for Centered PCA

The following theorem proves a regret bound for our Centered Online PCA Algorithm.

Theorem 8 *For any data sequence $x^1,\ldots,x^T$, initial center value m^0 such that $\|x^t - m^0\|_2 \le \frac{1}{2}$, any density matrix $U \in \mathcal{B}^n_{n-k}$ and any center vector m, the following bound holds:*

$$comp_{alg} \le \frac{\eta\, comp_{U,m} + \Delta(U,W^0) + \eta\widetilde{\eta}^{-1}(m-m^0)^\top U(m-m^0)}{1-\exp(-\eta)} + 1 + \log\big(1+\frac{T-1}{\widetilde{\eta}^{-1}+1}\big)$$

where

$$comp_{alg} = \sum_{i=1}^{T} \mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top)$$

is the overall expected compression loss of the centered online PCA Algorithm 6 and

$$comp_{U,m} = \sum_{i=1}^{T} \mathrm{tr}(U(x^t - m)(x^t - m)^\top)$$

is the total compression loss of comparison parameters (U,m).

Proof There are two main proof methods for the expert setting. The first is based on Bregman projections and was used so far in this paper. The second uses the value of the optimization problem defining the update as a potential and then shows that the drop of this value (Kivinen and Warmuth, 1999; Cesa-Bianchiand and Lugosi, 2006) is lower bounded by a constant times the per trial loss of the algorithm. Here we use a refinement of the second method that expresses the value of the optimization problem in terms of its dual. These variations of the second method were developed in the context of boosting (Warmuth et al., 2006; Liao, 2007) and in the conference paper (Kuzmin and Warmuth, 2007) where we enhanced the Uncentered Online PCA Algorithm of this paper with a kernel.

For our problem the value[6] of optimization problem (10) is $v^t = U^t(W^t, m^t)$ and this equals the value of the dual problem $\widehat{L}^t(\Gamma^t)$ where Γ^t maximizes the dual problem (13).

We want to establish the following key inequality:

$$v^t - v^{t-1} \;\ge\; \eta^{-1}(1-e^{-\eta})\Big(\mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top) - \frac{1}{\widetilde{\eta}^{-1}+t}\Big). \tag{17}$$

Since Γ^t optimizes the dual function $\widehat{L}^t$ and Γ^{t-1} is a non-optimal choice, $\widehat{L}^t(\Gamma^t) \ge \widehat{L}^t(\Gamma^{t-1})$ and therefore

$$v^t - v^{t-1} \;=\; \widehat{L}^t(\Gamma^t) - \widehat{L}^{t-1}(\Gamma^{t-1}) \;\ge\; \widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) \tag{18}$$

Substituting $\widehat{L}^t$ and $\widehat{L}^{t-1}$ from (13) into the right hand side of this inequality gives the following:

$$\begin{aligned}
&\widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) \\
&\quad = -\eta^{-1}\log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^t - \eta\Gamma^{t-1})) + \eta^{-1}\log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^{t-1} - \eta\Gamma^{t-1})) \\
&\quad = -\eta^{-1}\log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^t - \eta\Gamma^{t-1} - \log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^{t-1} - \eta\Gamma^{t-1}))).
\end{aligned}$$

Now we expand C^t and use the covariance matrix update from Lemma 7:

$$\begin{aligned}
\widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) = -\eta^{-1}\log\mathrm{tr}(\mathbf{exp}(&\log W^0 - \eta C^{t-1} - \eta\Gamma^{t-1} \\
&- \log\mathrm{tr}(\mathbf{exp}(\mathbf{log}\,W^0 - \eta C^{t-1} - \eta\Gamma^{t-1})) - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t - m^{t-1})(x^t - m^{t-1})^\top)).
\end{aligned}$$

6. Optimization problem (10) minimizes a convex function subject to linear cone constraint. Since this problem has a strictly feasible solution, strong duality is implied by a generalized Slater condition (Boyd and Vandenberghe, 2004).

The first four terms under the matrix exponential form $\log W^{t-1}$, which can be seen from the first expression for W^{t-1} from (14):

$$\widehat{L}^t(\Gamma^{t-1}) - \widehat{L}^{t-1}(\Gamma^{t-1}) = -\eta^{-1}\log \mathrm{tr}\Big(\mathbf{exp}(\mathbf{log}\, W^{t-1} - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t-m^{t-1})(x^t-m^{t-1})^\top)\Big).$$

Going back to (18) we get the inequality:

$$v^t - v^{t-1} \geq -\eta^{-1}\log \mathrm{tr}\Big(\mathbf{exp}(\mathbf{log}\, W^{t-1} - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t-m^{t-1})(x^t-m^{t-1})^\top)\Big).$$

This expression for the drop of the value is essentially the same expression that is normally bounded in the proof of online variance minimization algorithm in Warmuth and Kuzmin (2006a). Using those techniques (assumption in the theorem implies that that $\|x^t - m^{t-1}\|_2^2 \leq 1$ and all the necessary inequalities hold) we get the following inequality:

$$-\eta^{-1}\log \mathrm{tr}\Big(\mathbf{exp}(\mathbf{log}\, W^{t-1} - \eta\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(x^t-m^{t-1})(x^t-m^{t-1})^\top)\Big) \geq \eta^{-1}\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t}(1-e^{-\eta})\mathrm{tr}(W^{t-1}(x^t-m^{t-1})(x^t-m^{t-1})^\top).$$

W^{t-1} is a density matrix and its eigenvalues are at most 1. And by assumption, norm of $x^t - m^{t-1}$ is at most 1. Therefore, the loss $\mathrm{tr}(W^{t-1}(x^t-m^{t-1})(x^t-m^{t-1})^\top)$ is also at most 1. We split the factor in front of the loss as $\frac{\widetilde{\eta}^{-1}+t-1}{\widetilde{\eta}^{-1}+t} = 1 - \frac{1}{\widetilde{\eta}^{-1}+t}$, upper bounding the loss by 1 for the second part and leaving it as is for the first. With this (17) is obtained.

Note that the trace in the inequality (17) is the loss of the algorithm at trial t. Summation over t and telescoping gives us:

$$v^T - v^0 \geq \eta^{-1}(1-e^{-\eta})\Big(\mathrm{comp}_{\mathrm{alg}} - \sum_{t=1}^{T}\frac{1}{\widetilde{\eta}^{-1}+t}\Big).$$

We consider the left side first: v^0 is equal to zero, and v^T is a minimum of optimization problem (10), thus we can make it bigger by substituting arbitrary non-optimal values U and m. Index T means the optimization problem is defined with respect to the entire data sequence, therefore the loss term becomes the loss of the comparator. On the right side we use the following bound on the sum of generalized harmonic series: $\sum_{t=1}^{T}\frac{1}{\widetilde{\eta}^{-1}+t} \leq 1 + \log\Big(1+\frac{T-1}{\widetilde{\eta}^{-1}+1}\Big)$. Overall, we get:

$$\eta^{-1}\Delta(U,W^0) + \widetilde{\eta}^{-1}(m-m^0)^\top U(m-m^0) + \mathrm{comp}_{U,m} \geq \eta^{-1}(1-e^{-\eta})\left(\mathrm{comp}_{\mathrm{alg}} - \Big(1+\log\big(1+\frac{T-1}{\widetilde{\eta}^{-1}+1}\big)\Big)\right).$$

Moving things over and dividing results in the bound of the theorem. ■

As discussed before, when $\eta^{-1} = \widetilde{\eta}^{-1} = 0$, then the algorithm becomes the FL Algorithm. When $\widetilde{\eta}^{-1} \to \infty$, then m^t is clamped to m^0, that is, the update for the center (11,15) becomes $m^t = m^0$ and

is vacuous. Also in that case the term $\eta\widetilde{\eta}^{-1}(m-m^0)^\top U(m-m^0)$ in the upper bound of Theorem 8 is infinity unless the comparison center m is m^0 as well. If $m^0=0$ in addition to $\widetilde{\eta}^{-1}\to\infty$, then we call this the *uncentered version of Algorithm 6*: this version simply ignores step (15) and in (16) uses $m^{t-1}=0$. Our original Algorithm 5 for uncentered PCA as well as the uncentered version of Algorithm 6 have the same regret bound[7] of Theorem 6. Recall however that the two algorithms were motivated differently: Algorithm 5 trades off divergence to the last parameter with the loss in the last trial, whereas Algorithm 6 trades off a divergence to the initial parameter matrix with the total loss in all past trials. If all constraints are equality constraints, then the two algorithms are the same. However, capping introduces inequality constraints and therefore the two algorithms are decidedly not the same. Both algorithm can behave quite differently experimentally (Section 8). The difference between the two algorithms will become important in the followup paper (Kuzmin and Warmuth, 2007), where we were only able to use a kernel with the algorithm that trades off a divergence to the initial parameter matrix with the total loss in all past trials.

Similarly, when $\eta^{-1}\to\infty$, then W^t is clamped to W^0 and the algorithm degenerates to a previously analyzed algorithm, the Incremental Off-line Algorithm for Gaussian density estimation with fixed covariance matrix (Azoury and Warmuth, 2001). For this restricted density estimation problem, improved regret bounds were proven for the Forward Algorithm which further shrinks the estimate of the mean towards the initial mean. So far we were not able to improve our regret bound for uncentered PCA using additional shrinkage towards the initial mean.

The statement of the theorem requires strong initial knowledge about the center of the data sequence we are about to observe: the condition of the theorem says that our data sequence has to be contained in a ball of radius $\frac{1}{2}$ around m^0. This can be relaxed by using $m^0=0$ and $\widetilde{\eta}^{-1}=0$, which corresponds to using standard empirical mean for m^t. Now it suffices to assume that data is contained in some ball, but we are not required to know where exactly that ball is. The appropriate assumption and the change to the bound are detailed in the following corollary.

Corollary 9 *For any data sequence $x^1,\ldots,x^T$ that can be covered by a ball of radius $\frac{1}{2}$, that is, $\|x^{t_1}-x^{t_2}\|_2\le 1$ and that also has the bound on the norm of instances $\|x^t\|_2\le R$, any density matrix $U\in\mathcal{B}^n_{n-k}$ and any center vector m, the total expected loss of centered online PCA Algorithm 6 being used with parameters $\widetilde{\eta}^{-1}=0$ and $m^0=0$ is bounded as follows:*

$$comp_{alg}\le\frac{\eta\, comp_{U,m}+\Delta(U,W^0)}{1-e^{-\eta}}+\log T+R^2,$$

Proof The ball assumption means that the empirical mean m^{t-1} and any element of the data sequence are not too far from each other: $\|m^{t-1}-x^t\|_2\le 1$. Thus we can still use the Inequality (17), for all trials but the first one, where we haven't seen any data points yet. Summing the drops of the value starting from $t=1$ we get:

$$v^T-v^1\ge\eta^{-1}(1-e^{-\eta})\left(\sum_{t=2}^{T}\mathrm{tr}(W^{t-1}(x^t-m^{t-1})(x^t-m^{t-1})^\top)-\sum_{t=2}^{T}\frac{1}{t}\right).$$

7. The remaining $+1$ is an artifact of our bound on the harmonic sum.

We now add the loss of the first trial into the sum and rearrange terms:

$$\overbrace{\sum_{t=1}^{T} \mathrm{tr}(W^{t-1}(x^t - m^{t-1})(x^t - m^{t-1})^\top)}^{\mathrm{comp}_{\mathrm{alg}}}$$
$$\leq \frac{\eta(v^T - \overbrace{v^1}^{\geq 0})}{1-e^{-\eta}} + \overbrace{\sum_{t=2}^{T} \frac{1}{t}}^{\leq \log T} + \overbrace{\mathrm{tr}(W^0(x^1 - m^0)(x^1 - m^0)^\top)}^{\leq R^2}.$$

Finally, from the definition of v^T it follows that $v^T \leq \eta^{-1}\Delta(U, W^0) + \mathrm{comp}_{U,m}$, for any comparator U and m, and this gives the bound of the theorem. ■

Tuning η as in (3), Corollary 9 gives the following regret bound for our centered online PCA Algorithm 6 (when $k \leq \frac{n}{2}$):

(expected total compression loss of alg.) - (total comp. loss of best centered k-subspace)
$$\leq \sqrt{2(\text{total comp. loss of best centered } k\text{-subspace})\, k \log \frac{n}{k}} + k \log \frac{n}{k} + R^2 + \log T.$$

6.3 Converting the Online PCA Algorithms to Batch PCA Algorithms

In the online learning community a number of conversion techniques have been developed that allow one to construct a hypothesis with good generalization bounds in the batch setting from the hypotheses produced by a run of the online learning algorithm over the given batch of examples.

For example, using the standard conversion techniques developed for the expert setting based on the leave-one-out loss (Cesa-Bianchi et al., 1997), we obtain algorithms with good expected regret bounds in the following model: The algorithm is given $T-1$ instances drawn from a fixed but unknown distribution and produces a k-dimensional subspace based on those instances; it then receives a new instance from the same distribution. We can bound the expected loss on the new instance (under the usual norm less than one assumption on instances):

(expected compression loss of alg.) - (expected compression loss best k-space)
$$= O\Big(\sqrt{\frac{(\text{expected compression loss of best } k\text{-subspace})\, k \log \frac{n}{k}}{T}} + \frac{k \log \frac{n}{k}}{T}\Big).$$

The expected loss of the algorithm is taken as expectation over both the internal randomization of the algorithm and fixed distribution over the instances. The expected loss of the best subspace just averages over the distribution of the instances. The best subspace itself will be determined by the covariance matrix of this distribution.

Additionally, there also exist very general conversion methods that allow us to state bounds that say that the generalization error will be big with small probability (Cesa-Bianchi and Gentile, 2005). These bounds are more complicated and therefore we don't state them here. The conversion algorithms however, are pretty simple: for example, one can use the average density matrix of all density matrices produced by the online algorithm while doing one pass through the batch of instances. Perhaps surprisingly, the generalization bounds for batch PCA obtained via the online-to-batch conversions are competitive with the best bounds for batch PCA that we are aware of Shawe-Taylor et al. (2005).

7. Lower Bounds

We first prove some lower bounds for the simplest online algorithm that just predicts with the model that has incurred minimum loss so far (the Follow the Leader (FL) Algorithm). After that we give a lower bound for uncentered PCA that shows that the algorithm presented in this paper is optimal in a very strong sense.

Our first lower bound is in the standard expert setting. We assume that there is a deterministic tie-breaking rule, because by adding small perturbations, ties can always be avoided in this construction. It is easy to see that the following adversary strategy forces FL to have loss n times larger than the loss of the best expert chosen in hindsight: in each trial have the expert chosen by FL incur one unit of loss. Note that the algorithm incurs loss one in each trial, whereas the loss of the best expert is $\lfloor \frac{T}{n} \rfloor$ after T trials. We conclude that the loss of FL can be by a factor of n larger than the loss of the best expert.

We next show that for the set expert case, FL can be forced to have loss at least $\frac{n}{d}$ times the loss of the best set of size d. In this case FL chooses a set of size d of minimum loss and the adversary forces the lowest loss expert in the set chosen by FL to incur one unit of loss. The algorithm again incurs loss one in each trial, but the loss of the best set lies in the range $d\,[\lfloor \frac{T}{n} \rfloor, \lceil \frac{T}{n} \rceil]$. Thus in this case the loss of FL can be by a factor of $\frac{n}{d}$ larger than the loss of the best set of size d.

When rephrased i.t.o. compression losses, FL picks a set of size $n-d$ whose complementary set of size d has minimum compression loss. We just showed that the total compression loss of FL can be at least $\frac{n}{d}$ times the compression loss of the best subset of size $n-d$.

We can lift the above lower bound for sets to the case of uncentered PCA. Now $d = n-k$ and k is the rank of the subspace we want to compress onto. To simplify the argument, we let the first n instances be small multiples of the standard basis vectors. More precisely, $x_t = t\varepsilon\, e^t$, for $1 \leq t \leq n$ and small real ε. These instances cause the uncentered data covariance matrix $\sum_{t=1}^{n} x_t x_t^\top$ to be a diagonal matrix. Also, if ε is small enough then the loss in the first n trials is negligible. From now on FL always chooses a unique set of $d = n-k$ standard basis vectors of minimum loss and the adversary chooses a standard basis vector with the lowest loss in the set as the next instance. So the lower bound argument essentially reduces to the set case, and FL can be forced to have compression loss $\frac{n}{n-k}$ times the loss of the compression loss of the best k dimensional subspace.

So far we have shown that our online algorithms are better than the simplistic FL Algorithm since their compression losses are at most one times the loss of the best plus essentially a square root term. We now show that the constant in front of the square root term is rather tight as well. For the expert setting ($d = 1$) this was already done:

Theorem 10 *(Theorem C.3. of the journal version Helmbold and Warmuth 2008 of the conference paper Helmbold and Warmuth 2007.) For all $\varepsilon > 0$ there exists n_ε such that for any number of experts $n \geq n_\varepsilon$, there exists a $T_{\varepsilon,n}$ where for any number of trials $T \geq T_{\varepsilon,n}$ the following holds for any algorithm in the expert setting: there is a sequence of T trials with n experts for which the loss of the best expert is at most $T/2$ and the regret of the algorithm is at least $(1-\varepsilon)\sqrt{(T/2)\log n}$.*

In the expert model used in this paper, we follow Freund and Schapire (1997) and assume that the losses of the experts in each trial are specified by a loss vector in $[0,1]^N$. There is a related model (studied earlier), where the experts produce predictions in each trial. After receiving those predictions the algorithm produces its own prediction and receives a label. The loss of the experts

and algorithm is the absolute value of the difference between the predictions and the label, respectively (Littlestone and Warmuth, 1994). The above theorem actually holds for this model of online learning with the absolute loss (when all predictions are in $[0,1]$ and the labels are in $\{0,1\}$), and the model used in this paper may be seen as the special case where the prediction of the algorithm is formed by simply averaging the predictions of the experts (Freund and Schapire, 1997). Therefore any lower bound for the described expert model with absolute loss immediately holds for the expert model where the loss is specified by a loss vector.

Note that the regret bounds for the Hedge Algorithm discussed at the end of Section 3 have an additional factor of 2 in the square root term. By choosing a prediction function other than the weighted average, the factor of 2 can be avoided in the expert model with the absolute loss, and the upper and lower bounds for the regret have the same constant in front of the square root term (provided that N and T are large enough) (Cesa-Bianchi et al., 1997; Cesa-Bianchiand and Lugosi, 2006).

The above lower bound theorem immediately generalizes to the case of set experts. Partition the experts into d blocks of size $\frac{n}{d}$ (assume d divides n). For any algorithm and block, construct a sequence of length T as before. During the sequence for one block, the experts for all the other blocks have loss zero. The loss of the best set of size d on the whole sequence of length Td is at most $Td/2$ and the regret, that is, the loss of the algorithm on the sequence minus the loss of the best set of size d, is lower bounded by

$$(1-\varepsilon)d\sqrt{\frac{T}{2}\log\frac{n}{d}} = (1-\varepsilon)\sqrt{\frac{dT}{2}d\log\frac{n}{d}}.$$

Rewritten in terms of compression losses for compression sets of size k (i.e., $d = n-k$), the lower bound on the compression loss regret becomes

$$(1-\varepsilon)\sqrt{\text{compression loss of best } k\text{-subset } (n-k)\log\frac{n}{n-k}}. \tag{19}$$

Note that the upper bound (4) obtained by our algorithm for learning as well as the best subset is essentially a factor of $\sqrt{2}$ larger than this lower bound.

Finally, we lift the above lower bound for subsets to a lower bound for uncentered PCA. In the setup for uncentered PCA, the instance matrix at trial t is $S^t = x^t(x^t)^\top$, where x^t has 2-norm at most 1. For the lower bound we need the instance matrix S^t to be an arbitrary symmetric matrix with eigenvalues in $[0,1]$. As discussed before Section 5.1, the upper bound for uncentered PCA still holds for these more general instance matrices.

To lift the lower bound for subsets to uncentered PCA, we simply replace the loss vector ℓ^t by the instance matrix $S^t = \operatorname{diag}(\ell^t)$. At trial t, the PCA algorithm uses the density matrix W^{t-1} and incurs expected loss $\operatorname{tr}(W^{t-1}\operatorname{diag}(\ell^t)) = \operatorname{diag}(W^{t-1})\cdot\ell^t$. Note that the diagonal vector $\operatorname{diag}(W^{t-1})$ is a probability vector. Thus the PCA algorithm doesn't have any advantage from using non-diagonal density matrices and the lower bound reduces to the set case. We conclude that the lower bound (19) also holds for the compression loss regret of uncentered PCA algorithms when the instance matrices are allowed to be symmetric matrices with eigenvalues in $[0,1]$. Again the corresponding upper bound (7) is essentially a factor of $\sqrt{2}$ larger.

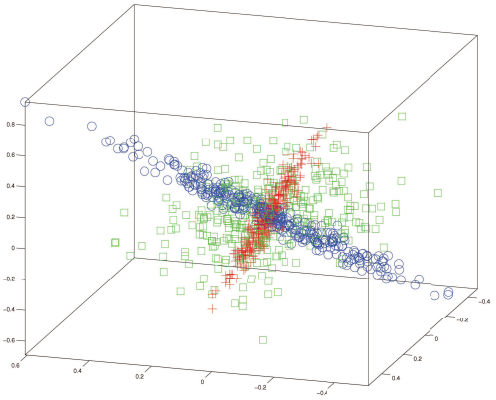


Figure 3: The data sequence used for the first experiment switches between three different subspaces. It is split into three segments. Within each segment, the data is drawn from a different 20-dimensional Gaussian with a rank 2 covariance matrix. We plot the first three coordinates of each data point. Different colors/symbols denote the data points that came from the three different subspaces.

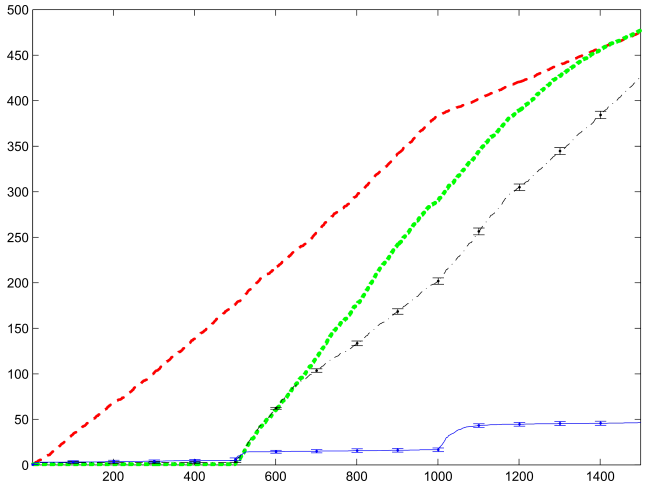


Figure 4: The blue/solid curve is the total loss of uncentered online PCA Algorithm 5 for the data sequence described in Figure 3 (with $n = 20, k = 2$ and $\eta = 1$). The algorithm uses internal randomization for choosing a subspace and therefore the curve is actually the average total loss over 50 runs for the same data sequence. The error bars (one standard deviation) indicate the variance of the algorithm. The black/dash-dotted curve plots the same for the uncentered version of Algorithm 6 (again $\eta = 1$). The visible bumps in the curves correspond to places in the data sequence where it shifts from one subspace to another. The red/dashed curve is the total loss of the best projection matrix determined in hindsight (i.e., loss of batch uncentered PCA). The green/dotted curve is the total loss of the Follow the Leader Algorithm.

8. Simple Experiments

The regret bounds we prove for our online PCA algorithms hold for arbitrary sequences of instances. In other words, they hold even if the instances are produced by an adversary which aims to make the algorithm have large regret. In many cases, natural data does not have a strong adversarial nature and even the simple Follow the Leader Algorithm might have small regret against the best subspace chosen in hindsight. However, natural data commonly shifts with time. It is on such time-changing data sets that online algorithms have an advantage. In this section we present some simple experiments that bring out the ability of our online algorithms to adapt to data that shifts with time.

For our first experiment we constructed a simple synthetic data set of time-changing nature. The data sequence is divided into three equal intervals, of 500 points each. Within each interval data points are picked at random from a multivariate Gaussian distribution on $\mathbb{R}^{20}$ with zero mean.

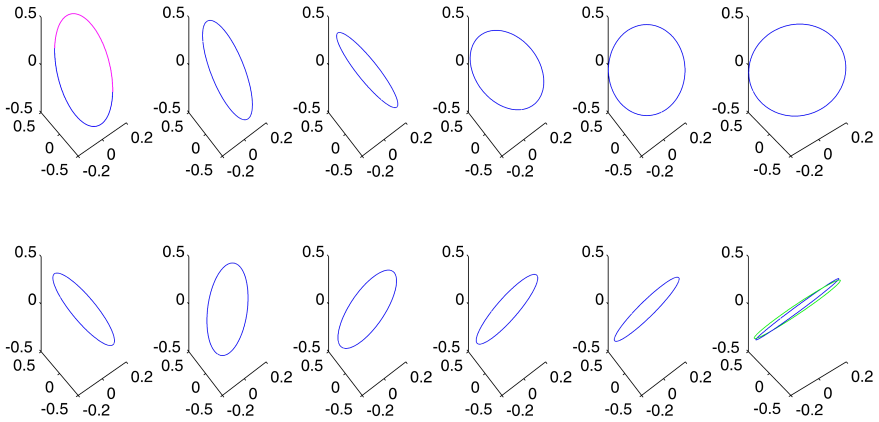


Figure 5: Behavior of the Uncentered Online PCA Algorithm 5 when data shifts from one subspace to another. First shift for one of the runs in Figure 4 is shown. We show the projection matrices that have the highest probability of being picked by the algorithm in a given trial. Since $k = 2$, each such matrix P^t can be seen as a 2-dimensional ellipse in $\mathbb{R}^{20}$: the ellipse is formed by points $P^t x$ for all $\|x\|_2 = 1$. We plot the first three coordinates of this ellipse. The transition sequence starts with the algorithm focused on the optimal projection matrix for the first subset of data and ends with essentially the optimal matrix for the second subset. The depicted transition takes about 60 trials and only every $5th$ trial is plotted.

The covariance matrices for the Gaussians were picked at random but constrained to rank 2, thus ensuring that the generated points lie in some 2-dimensional subspace. The generated points with norm bigger than one were normalized to 1. The data set is graphically represented in Figure 3, which plots the first three dimensions of each one of the data points. Different colors/symbols indicate data points that came from the three different subspaces.

In Figure 4 we plot the total compression loss for some of the algorithms introduced in this paper. For the sake of simplicity we restrict ourselves to uncentered PCA. Here the data dimension n is 20 and the subspace dimension k is 2. We plot the total loss of the following algorithms as a function of the trial number: the FL Algorithm, the original uncentered PCA Algorithm 5 and the uncentered version of Algorithm 6. For the latter two algorithms we need to select a learning rate. One possibility is to choose the learning rate that optimizes our upper bound on the regret of the algorithms (3). Since the bound is the same for both algorithms this choice of η is also the same: the choice depends on an upper bound on the compression loss of the batch algorithm. Plugging in the actual compression loss of the batch algorithm gives $\eta = 0.12$. In practice, heuristics can be used to tune η. For the experiment of Figure 4 we simply chose $\eta = 1$. Recall that our online PCA algorithms decompose their density matrix parameter W^t into a convex combination of projection matrices using the deterministic Algorithm 2. However, the PCA algorithms then randomly select one of the k dimensional projection matrices in the convex combination with probability equal to its coefficient. This introduces randomness into the execution of the algorithm even when run on a fixed data sequence. We run the algorithms 50 times and plot the average total loss as a function

of t for the fixed data sequence depicted in Figure 3. We also indicate the variance of this total loss with error bars of one standard deviation. Note again that the average and variance is w.r.t. internal randomization of the algorithm. The bumps in the loss curves of Figure 4 correspond to the places in the data sequence where it shifts from one subspace to another. When the loss curve of an algorithm flattens out then the algorithm has learned the correct subspace for the current segment. For example, Figure 5 depicts how the density matrix of the uncentered PCA Algorithm 5 ($\eta = 1$) transitions around a segment boundary.

The FL Algorithm (which coincides with the uncentered version of Algorithm 10 when $\eta \to \infty$) learns the first segment really quickly, but it does not recover during the later segments. The original uncentered PCA Algorithm 5 (with $\eta = 1$) recovers in each segment, whereas the uncentered version of Algorithm 6 (with $\eta = 1$) does not recover as quickly and has higher total loss on this data sequence. Recall that both online algorithms for uncentered PCA have the same regret bound against the best fixed offline comparator.

In Figure 4 we also plot the compression loss of the best subspace selected in hindsight by running batch uncentered PCA (dashed/red line). The data set we generated is not well approximated by a single 2-dimensional subspace, since it consists of three different 2-dimensional subspaces. As a result, the overall loss of the fixed subspace is higher than the loss of both of our uncentered online PCA algorithms that to a varying extent were able to switch between the different 2-dimensional subspaces.

There are many heuristics for detecting switches of the data. For example one could simply check for a performance loss of any algorithm in a suitably chosen final segment. However, such algorithms are often unwieldy and are hard to tune when the data in difference segments are not drastically different. Note that for the Uncentered Online PCA Algorithm presented in this paper we only have provable good regret bounds against the best fixed subspace chosen offline (based on the entire data sequence). (In Figure 4 both of our online algorithms beat the offline comparator and thus have negative regret against this simple comparator.) Ideally we would like measure regret against stronger offline comparators that can exploit the shifting nature of the data. In the expert setting there is a long line of research where the offline algorithm is allowed to partition the data sequence into s segments and pick the best subspace for each of the s segments. There are simple modifications of the online algorithm in the experts setting that have good regret bounds against the best partition of size s (see, e.g., Herbster and Warmuth, 1998). Naturally, the regret bounds grow with the number of segments s. The modifications that are needed to achieve these regret bounds are simple: insert an additional update step at the end of each trial that mixes a little bit of the uniform distribution into the weight vector. In the context of uncentered PCA this amounts to adding the following update at the end of each trial:

$$W^t = (1-\alpha)W^t + \alpha\frac{I}{n}.$$

We claim that it is easy to again lift the techniques developed in the expert setting to PCA and prove the analogous regret bounds against the best partition.

The following subtle modification of the mixing rule leads to algorithms with even more interesting properties. If the algorithm mixes in a little bit of the average of all past weight vectors instead of the uniform vector, then it is able to switch quickly to experts that have been good at some time in the past. This has been called the "long-term memory" property of such algorithms. In the context of uncentered PCA, this amounts the following additional update step for the density

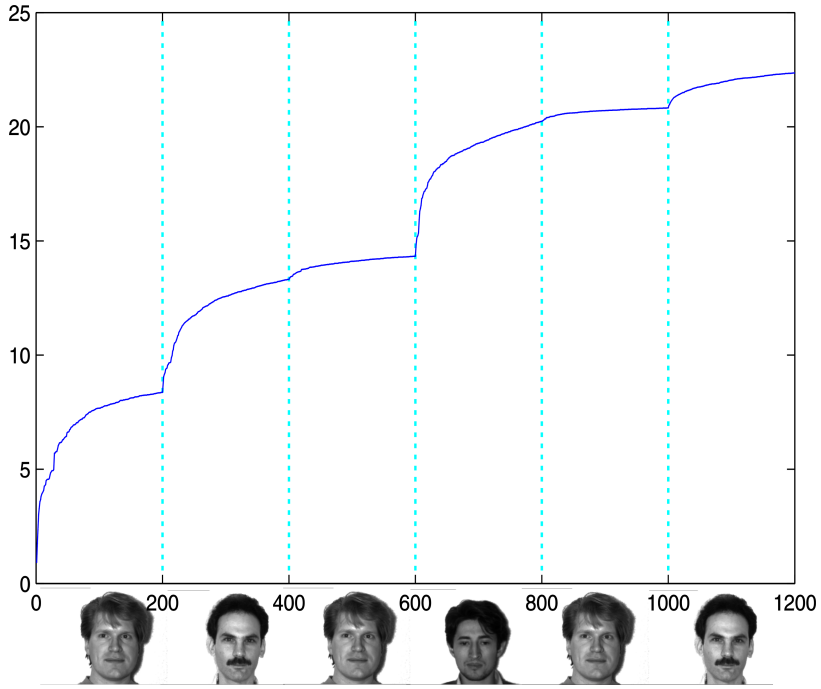


Figure 6: Long term memory effect when mixing in past average is added to the uncentered online PCA Algorithm 5. The data sequence is comprised of several segments, each one of two hundred randomly sampled images of the same person but with different facial expressions and lighting conditions. These segments are indicated with dotted lines and are labeled with the face of the person that was used to generate the segment. The plot depicts the regret of the uncentered online PCA Algorithm 5 ($\eta = 1$) with added mixing update (20) ($\alpha = .001$). The regret is w.r.t. the indicated partition of size six.

matrix:

$$W^t = (1-\alpha)W^t + \alpha\frac{\sum_{q=0}^{t-1} W^q}{t}. \tag{20}$$

We performed another experiment that demonstrates this "long-term memory" effect by adding update (20) to the uncentered online PCA Algorithm 5: For this experiment we used face image data from Yale-B data set. A segmented data sequence was constructed by sampling the face images, where each segment contained images of the same person, but with different facial expressions, lighting conditions, etc. Figure 6 plots the regret of our uncentered online PCA Algorithm 5 against the best partition of size 6 chosen offline ($\eta = 1, \alpha = .001$). In the picture the segment boundaries are indicated with dotted lines and the face below each segment shows the person who's pictures were sampled during that section. Note that our online algorithm doesn't have knowledge of segment boundaries. The first segment shows the typical behavior of online algorithms: the regret grows initially, but then flattens out once the algorithm converges to the best solution for that segment. Thus the "bump" in the regret curve is due to the fact that the algorithm has to learn a new segment of pictures from a different person. By comparing the bumps of different segments we see that they are significantly smaller in segments where the algorithm encounters pictures from a person that it

has previously seen: the algorithm "remembers" subspaces that were good in the past. These small bumps can be seen in segments 3,5 and 6 in our data set. For additional plots of long-term memory effects in the expert setting see Bousquet and Warmuth (2002). Our experiments are preliminary and we did not formally prove regret bounds for shifting comparators in PCA setting, but such bounds are straightforwardly obtained using the methodology from this paper and Bousquet and Warmuth (2002).

9. Conclusions

We developed a new set of techniques for learning as well as a low dimensional subspace. We first developed the algorithms in the expert case and then lifted these algorithms and bounds to the matrix case as essentially done in Tsuda et al. (2005); Warmuth and Kuzmin (2006a). The new insight is to represent our uncertainty over the subspace as a density matrix with capped eigenvalues. We show how to decompose such a matrix into a mixture of n subspaces of the desired rank. In the case of PCA the seemingly quadratic compression loss can be rewritten as a linear loss of our matrix parameter. Therefore a random subspace chosen from the mixture has the same expected loss as the loss of the parameter matrix.

Similar techniques (albeit not capping) have been used recently for learning permutations (Helmbold and Warmuth, 2008): Instead of maintaining a probability vector over the $n!$ permutations, the parameter used in that paper is a convex combination of permutation matrices which is an $n \times n$ doubly stochastic matrix. This matrix can be efficiently decomposed into a mixture of $O(n^2)$ permutation matrices. If the loss is linear, then a random permutation chosen from the mixture has the same expected loss as the loss of the doubly stochastic parameter matrix.

When the loss is convex and not linear (as for example the quadratic loss in the generalization of linear regression to matrix parameters Tsuda et al. 2005), then our new capping trick can still be applied to the density matrix parameter. In this case, the online loss of the capped density matrix can be shown to be close to the loss of the best low dimensional subspace. However the quadratic loss of the capped density matrix (which is a convex combination of subspaces) can be much smaller than the convex combination of the losses of the subspaces. The bounds of that paper only hold for the smaller loss of the capped density matrix and not for the expected loss of the subspace sampled from the convex combination represented by the capped density matrix.

Our new capping technique is interesting in its own right. Whereas the vanilla multiplicative update on n experts with exponential factors is prone to be unstable because it tends to go into a corner of the simplex (by putting all weight on the currently best expert), the technique of capping the weight at $\frac{1}{d}$ keeps a set of at least d experts alive. In some sense the capping has the same effect as a "super predator" has on a biological system: such a predator specializes on the most frequent species of prey, preventing the dominance of any particular species and thus preserving variety. See Warmuth (2007a) for a discussion of the relationships of our updates to biological systems.

Following Kivinen and Warmuth (1997); Helmbold et al. (1999), we motivate our updates by trading off a parameter divergence against a loss divergence. In our case, the parameter divergence is always a capped relative entropy and the loss divergence is simply linear. It would be interesting to apply the capping trick to the logistic loss used in logistic regression. The logistic loss can be seen as a relative entropy between the desired probabilities of the outcomes and the predicted probabilities of the outcomes obtained by applying a sigmoid function to the linear activations (Kivinen and Warmuth, 2001). For "capped logistic regression", linear constraints would be added to the op-

timization problem defining the updates that cap the predicted probabilities of the outcomes. This would lead to versions of logistic regression where the loss favors a small set of outcomes instead of a single outcome.

The algorithms of this paper belong to the family of multiplicative updates, that is, the parameters are updated by multiplicative factors of exponentials. In the matrix case the updates make use of the matrix log and matrix exponential. There is a second family of updates that does additive parameter updates. In particular, there are additive online updates for PCA (Crammer, 2006). The latter update family has the key advantage that they can be easily used with kernels. However, in a recent conference paper (Kuzmin and Warmuth, 2007) we also were able to give a special case where the multiplicative updates could be enhanced with a kernel. In this case the instance matrices are outer products $x^t(x^t)^\top$ and are replaced by $\phi(x)^t(\phi(x^t))^\top$. In particular, the online PCA algorithms of this paper can be "kernelized" this way.

In PCA the instance matrices are *symmetric* matrices $x^t(x^t)^\top$ and we seek a low rank *symmetric* subspace that approximates the instances well. In a recent conference paper we showed in a slightly different context how to generalize our methods to the case of "asymmetric" matrices of arbitrary shape. Using those techniques, the online PCA algorithms of this paper can be generalized to the asymmetric case: now the instance matrices are rank one $n \times m$ matrices of the form $x^t(\widetilde{x}^t)^\top$, where x and $\widetilde{x}$ are vectors of dimension n and m, respectively. In the asymmetric case, the underlying decomposition is the SVD decomposition instead of the eigendecomposition.

There are two technical open problems that arise in this paper. We gave a number of dynamic programming algorithms, such as the one given for the set expert problem. If the loss range for the individual experts is [0,1] then the loss range for the set experts is $[0,d]$ when d is the size of the set. The straightforward application of results for the Hedge Algorithm leads to extra factors of d in the regret bounds for set experts. We avoided these factors using our capping trick. However, the question is whether the same bounds (without the d factors) hold for the dynamic programming algorithm as well. There is a different fundamental algorithmic technique for dealing with structural domains: the Follow the Perturbed Leader Algorithm (Kalai and Vempala, 2005). The second open problem is how to adapt this algorithm to online PCA and prove bounds for it that don't contain the extra d factors.

Finally, independent of what theoretical progress might have been achieved in this paper, we still have to find a convincing experimental setting where our new online PCA algorithms are indispensable. The update time of our algorithms is $O(n^2)$ and further time improvements via approximations or lazy evaluation might be needed to make the algorithms widely applicable.

Acknowledgments

Thanks to Allen Van Gelder for valuable discussions re. Algorithm 2.

References

K. Azoury and M. K. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Journal of Machine Learning*, 43(3):211–246, June 2001. Special issue on *Theoretical Advances in On-line Learning, Game Theory and Boosting*, edited by Yoram Singer.

R. Bhatia. *Matrix Analysis*. Springer, Berlin, 1997.

O. Bousquet and M. K. Warmuth. Tracking a small set of experts by mixing past posteriors. *Journal of Machine Learning Research*, 3:363–396, 2002.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.

Y. Censor and A. Lent. An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, 34(3):321–353, July 1981.

N. Cesa-Bianchi and C. Gentile. Improved risk tail bounds for on-line algorithms. In *NIPS*, 2005.

N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, May 1997.

N. Cesa-Bianchiand and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

K. Crammer. Online tracking of linear subspaces. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 06)*, Pittsburg, June 2006. Springer.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.

M. Gu and S. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1266–1276, October 1994.

D. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. In *Proceedings of the 20th Annual Conference on Learning Theory (COLT07)*. Springer, 2007.

D. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. Submitted journal version, August 2008.

D. P. Helmbold, J. Kivinen, and M. K. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, November 1999.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998. Earlier version in 12th ICML, 1995.

M. Herbster and M. K. Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1:281–309, 2001.

A. Kalai. Simulating weighted majority with FPL. Private communication, 2005.

A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, 2005. Special issue Learning Theory 2003.

J. Kivinen and M. K. Warmuth. Averaging expert predictions. In *Computational Learning Theory, 4th European Conference, EuroCOLT '99, Nordkirchen, Germany, March 29-31, 1999, Proceedings*, volume 1572 of *Lecture Notes in Artificial Intelligence*, pages 153–167. Springer, 1999.

J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, 132(1):1–64, January 1997.

J. Kivinen and M. K. Warmuth. Relative loss bounds for multidimensional regression problems. *Machine Learning*, 45(3):301–329, 2001.

J. Kivinen, M. K. Warmuth, and B. Hassibi. The *p*-norm generalization of the LMS algorithm for adaptive filtering. *Journal of IEEE Transactions on Signal Processing (to appear)*, 54(5): 1782–1793, May 2005.

D. Kuzmin and M. K. Warmuth. Optimum follow the leader algorithm. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT 05)*, pages 684–686. Springer, June 2005. Open problem.

D. Kuzmin and M. K. Warmuth. Online Kernel PCA with entropic matrix updates. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*. ACM Press, June 2007.

J. Liao. *Totally Corrective Boosting Algorithms that Maximize the Margin*. PhD thesis, University of California at Santa Cruz, June 2007.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Inform. Comput.*, 108(2): 212–261, 1994. Preliminary version in in FOCS 89.

A. W. Marshall and I. Olkin. *Inequalities: Theory of Majorization and its Applications*. Academic Press, 1979.

R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

T. D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473, 1989.

J. Shawe-Taylor, C. K. I. Williams, N. Cristianini, and J. S. Kandola. On the eigenspectrum of the gram matrix and the generalization error of kernel-PCA. *IEEE Transactions on Information Theory*, 51(7):2510–2522, 2005. URL `http://dx.doi.org/10.1109/TIT.2005.850052`.

E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4:773–818, 2003.

K. Tsuda, G. Rätsch, and M. K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projections. *Journal of Machine Learning Research*, 6:995–1018, June 2005.

M. K. Warmuth. The blessing and the curse of the multiplicative updates. Work in progress, unpublished manuscript., 2007a.

M. K. Warmuth. When is there a free matrix lunch. In *Proc. of the 20th Annual Conference on Learning Theory (COLT 07)*. Springer, June 2007b. Open problem.

M. K. Warmuth and D. Kuzmin. Online variance minimization. In *Proceedings of the 19th Annual Conference on Learning Theory (COLT 06)*, Pittsburg, June 2006a. Springer.

M. K. Warmuth and D. Kuzmin. Randomized PCA algorithms with regret bounds that are logarithmic in the dimension. In *Advances in Neural Information Processing Systems 19 (NIPS 06)*. MIT Press, December 2006b.

M. K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*, pages 1001–1008, New York, NY, USA, 2006. ACM Press.

Journal of Machine Learning Research 9 (2008) 2321-2348 Submitted 6/07; Revised 1/08; Published 10/08

Probabilistic Characterization of Random Decision Trees

Amit Dhurandhar ASD@CISE.UFL.EDU
Alin Dobra ADOBRA@CISE.UFL.EDU
Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611, USA

Editor: Greg Ridgeway

Abstract

In this paper we use the methodology introduced by Dhurandhar and Dobra (2009) for analyzing the error of classifiers and the model selection measures, to analyze decision tree algorithms. The methodology consists of obtaining parametric expressions for the moments of the generalization error (GE) for the classification model of interest, followed by plotting these expressions for interpretability. The major challenge in applying the methodology to decision trees, the main theme of this work, is customizing the generic expressions for the moments of GE to this particular classification algorithm. The specific contributions we make in this paper are: (a) we primarily characterize a subclass of decision trees namely, Random decision trees, (b) we discuss how the analysis extends to other decision tree algorithms and (c) in order to extend the analysis to certain model selection measures, we generalize the relationships between the moments of GE and moments of the model selection measures given in (Dhurandhar and Dobra, 2009) to randomized classification algorithms. An empirical comparison of the proposed method with Monte Carlo and distribution free bounds obtained using Breiman's formula, depicts the advantages of the method in terms of running time and accuracy. It thus showcases the use of the deployed methodology as an exploratory tool to study learning algorithms.

Keywords: moments, generalization error, decision trees

1. Introduction

Consider the problem of estimating how a given classification algorithm (rather than a particular classifier) performs on a given joint distribution over the input-output space ($X \times Y$). As opposed to the general setup in machine learning where the distribution is unknown and only independent and identically distributed (i.i.d.) samples are available, in this scenario, *in principle*, the behavior of classification algorithm can be accurately studied. If this problem be solved efficiently, it offers an alternative line of study for classification algorithms and potentially unique insights into the *non-asymptotic* behavior of learning algorithms.

While the problem of estimating classification algorithm performance on a given distribution might look simple, solving it efficiently poses significant technical hurdles. The most natural way of studying a classification algorithm would be to sample N datapoints from the given distribution, train the algorithm to produce a classifier, test the classifier on a few sampled test sets and report the average error computed over these test sets. A shortcoming of the above approach is that based on just one single instance of the algorithm (since the algorithm was trained on a single data set of size N) we conclude about its general behavior. A straightforward extension of the above approach

to make the results more relevant in studying the algorithm would be to sample multiple data sets of size N, train on each of them to produce different classifiers, compute the test error for each of the classifiers and calculate the average and variance of the obtained test errors. This procedure would be a better indicator of the behavior of the algorithm than the previous case since we study multiple instances of the algorithm than just an isolated instance. Ideally, we would want to study the behavior of the algorithm by training it on all possible data sets of size N producing a variety of classifiers and then evaluating the expected value and variance of the generalization error (GE) of each of these classifiers. The GE of a classifier ζ is given by,

$$\begin{aligned} GE(\zeta) &= E\left[\lambda(\zeta(x),y)\right] \\ &= P\left[\zeta(x) \neq y\right] \end{aligned}$$

where $\lambda(.,.)$ is a 0-1 loss function, x is an input and y is an output and the expectation is over the input-output space $X \times Y$. The expected value and variance of GE over all possible classifiers[1] are denoted by,

$$E_{\mathcal{Z}(N)}\left[GE(\zeta)\right],$$

$$Var(GE(\zeta)) = E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[GE(\zeta)GE(\zeta')\right] - E_{\mathcal{Z}(N)}\left[GE(\zeta)\right]^2$$

where $\mathcal{Z}(N)$ represents the space of all possible classifiers produced by training the classification algorithm on all data sets of size N (denoted by $D(N)$), drawn from the joint distribution. With this we have shown that the moments provide a natural and informative avenue for studying classification algorithms. The question that now arises is, can we compute them efficiently. In our previous work (Dhurandhar and Dobra, 2009), we presented a general framework for computing these quantities for an arbitrary classification algorithm efficiently. By extensive use of the linearity of expectation and change of the order of sums (and integrals), the moments of GE can be expressed in terms of the behavior of the classification algorithm on specific inputs rather than on the whole space, thus reducing the complexity from an exponential in the size of the input space to linear for the computation of the first moment and quadratic for the second moment. As part of this prior work, the generic expressions to compute the moments were customized for the Naive Bayes Classification algorithm. In the present work we customize the generic expressions to compute moments of the generalization error for a more popular classification algorithm: Random decision trees.

The specific contributions we make are: We develop a characterization for a subclass of decision trees. In particular, we characterize Random decision trees which are an interesting variant with respect to three popular stopping criteria namely; fixed height, purity and scarcity (i.e., fewer than some threshold number of points in a portion of the tree). The analysis directly applies to categorical as well as continuous attributes with split points predetermined for each attribute. Moreover, the analysis in Section 3.3 is applicable to even other deterministic attribute selection methods based on information gain, gini gain etc. These and other extensions of the analysis to continuous attributes with dynamically chosen split points is discussed in Section 5. In the experiments that ensue the theory, we compare the accuracy of the derived expressions with direct Monte Carlo (i.e., hold-out-set estimation) and Breiman's strength and correlation based bounds (Breiman, 2001) on synthetic

1. Expectations over $\mathcal{Z}(N)$ are more general than over $D(N)$ since the classification algorithm can be randomized.

distributions as well as on distributions built on real data. Notice that using the expressions, the moments can be computed without explicitly building the tree. We also extend the relationships between the moments of GE and moments of cross validation error (CE), leave-one-out error (LE) and hold-out-set error (HE) given in Dhurandhar and Dobra (2009) which were applicable only to deterministic classification algorithms, to be made applicable to randomized classification algorithms.

2. Preliminaries

Model selection for classification is one of the major challenges in Machine Learning and Datamining. Given an i.i.d. sample from the underlying probability distribution, the classification model selection problem consists in building a classifier by selecting among competing models. Ideally the model selected minimizes GE. Since GE cannot be directly computed, part of the sample is used to estimate GE through measures such as cross validation, hold-out-set, leave-one-out, etc. Though certain rules of thumb are followed by practitioners w.r.t. training size and other parameters specific to the validation measures in evaluating models through empirical studies (Kohavi, 1995; Blum et al., 1999) and certain asymptotic results exist (Vapnik, 1998; Shao, 1993), the fact remains that most of these models and model selection measures are not well understood in real life (non-asymptotic) scenarios (e.g., what fraction should be test and training, what should be the value k in k-fold cross validation etc.). This lack of deep understanding limits our ability of using the models most effectively and maybe more importantly trusting the models to perform well in a particular application.

Recently, a novel methodology was proposed in Dhurandhar and Dobra (2009) to study the behavior of models and model selection measures. Since the methodology is at the core of the current work, we briefly describe it together with the motivation for using this type of analysis for classification in general and decision trees in particular.

2.1 *What* is the Methodology?

The methodology for studying classification models consists of studying the behavior of the first two central moments of the GE of the classification algorithm studied. The moments are taken over the space of all possible classifiers produced by the classification algorithm, by training it over all possible data sets sampled i.i.d. from some distribution. The first two moments give enough information about the statistical behavior of the classification algorithm to allow interesting observations about the behavior/trends of the classification algorithm w.r.t. any chosen data distribution.

2.2 *Why* have such a Methodology?

The answers to the following questions shed light on why the methodology is necessary if tight statistical characterization is to be provided for classification algorithms.

1. *Why study GE ?* The biggest danger of learning is *overfitting* the training data. The main idea in using GE as a measure of success of learning instead on the empirical error on a given data set is to provide a mechanism to avoid this pitfall. Implicitly, by analyzing GE all the input is considered.

2. *Why study the moments instead of the distribution of GE ?* Ideally, we would study the distribution of GE instead of moments in order to get a complete picture of what is its behavior.

Studying the distribution of discrete random variables, except for very simple cases, turns out to be very hard. The difficulty comes from the fact that even computing the probability of a single point is intractable since all combinations of random choices that result in the same value for GE have to be enumerated. On the other hand, the first two central moments coupled with distribution independent bounds such as Chebychev and Chernoff give guarantees about the worst possible behavior that are not too far from the actual behavior (small constant factor). Interestingly, it is possible to compute the moments of a random variable like GE without ever explicitly writing or making use of the formula for the cumulative distribution function. What makes such an endeavor possible is extensive use of the linearity of expectation.

3. *Why characterize a class of classifiers instead of a single classifier ?* While the use of GE as the success measure is standard practice in Machine Learning, characterizing classes of classifiers instead of the particular classifier produced on a given data set is not. From the point of view of the analysis, without large testing data sets it is not possible to evaluate directly GE for a particular classifier. By considering classes of classifiers to which a classifier belongs, an indirect characterization is obtained for the particular classifier. This is precisely what Statistical Learning Theory (SLT) does; there the class of classifiers consists in all classifiers with the same VC dimension. The main problem with SLT results is that classes based on VC dimension are too large, thus results tend to be pessimistic. In our methodology, the class of classifiers consists only of the classifiers that are produced by the given classification algorithm from data sets of fixed size from the underlying distribution. This is the probabilistic smallest class in which the particular classifier produced on a given data set can be placed in.

2.3 *How* do we Implement the Methodology ?

One way of approximately estimating the moments of GE over all possible classifiers for a particular classification algorithm is by directly using Monte Carlo. If we use Monte Carlo directly, we first need to produce a classifier on a sampled data set then test on a number of test sets sampled from the same distribution acquiring an estimate of the GE of this classifier. Repeating this entire procedure a few times we would acquire estimates of GE for different classifiers. Then by averaging the error of these multiple classifiers we would get an estimate of the first moment of GE. The variance of GE can also be similarly estimated. The problem with this procedure is that the space of all possible data sets can be huge. For instance, if we have d attributes each taking m values then the number of possible data sets of size N is $N^{m^d} - 1$. Even for any reasonable assignment to N (say, 100), m (say 2) and d (say 3) the number of experiments that need to be performed to guarantee accurate (if not exact) estimation of the moments seems unreasonable.

Another way of estimating the moments of GE, is by obtaining parametric expressions for them. If this can be accomplished the moments can be computed exactly. Moreover, by dexterously observing the manner in which expressions are derived for a particular classification algorithm, insights can be gained into analyzing other algorithms of interest. Though deriving the expressions may be a tedious task, using them we obtain highly accurate estimates of the moments. In this paper, we propose this second alternative for analyzing a subclass of decision trees. The key to the analysis is focusing on the learning phase of the algorithm. In cases where the parametric expressions are computationally intensive to compute directly, we show that approximating individual terms using Monte Carlo we obtain accurate estimates of the moments when compared to directly using Monte Carlo (first alternative) for the same computational cost.

If the moments are to be studied on synthetic data then the distribution is anyway assumed and the parametric expressions can be directly used. If we have real data an empirical distribution can be built on the data set and then the parametric expressions can be used.

2.4 Applications of the Methodology

It is important to note that the methodology is not aimed towards providing a way of estimating bounds for GE of a classifier on a given data set (i.e., finding distribution free bounds). The primary goal is creating an avenue in which learning algorithms can be studied precisely, that is, studying the statistical behavior of a particular algorithm w.r.t. a chosen/built distribution. Below, we discuss the two most important perspectives in which the methodology can be applied.

2.4.1 ALGORITHMIC PERSPECTIVE

If a researcher/practitioner designs a new classification algorithm, he/she needs to validate it. Standard practice is to validate the algorithm on a relatively small (5-20) number of data sets and to report the performance. By observing the behavior of only a few instances of the algorithm the designer infers its quality. Moreover, if the algorithm under performs on some data sets, it can be sometimes difficult to pinpoint the precise reason for its failure. If instead he/she is able to derive parametric expressions for the moments of GE, the test results would be more relevant to the particular classification algorithm, since the moments are over all possible data sets of a particular size drawn i.i.d. from some chosen/built distribution. Testing individually on all these data sets is an impossible task. Thus, by computing the moments using the parametric expressions the algorithm would be tested on a plethora of data sets with the results being highly accurate. Moreover, since the testing is done in a controlled environment, that is, all the parameters are known to the designer while testing, he/she can precisely pinpoint the conditions under which the algorithm performs well and the conditions under which the algorithm under performs.

2.4.2 DATA SET PERSPECTIVE

If an algorithm designer validates his/her algorithm by computing moments as mentioned earlier, it can instill greater confidence in the practitioner searching for an appropriate algorithm for his/her data set. The reason for this being, if the practitioner has a data set which has a similar structure or is from a similar source as the test data set on which an empirical distribution was built and favorable results reported by the designer, then this would mean that the results apply not only to that particular test data set, but to other similar type of data sets and since the practitioner's data set belongs to this similar collection, the results would also apply to his. Note that a distribution is just a weighting of different data sets and this perspective is used in the above exposition.

3. Computing Moments

In this section we first provide the necessary technical groundwork, followed by customization of the expressions for decision trees. We now introduce some notation that is used primarily in this section. X is a random vector modeling input whose domain is denoted by $\mathcal{X}$. Y is a random variable modeling output whose domain is denoted by $\mathcal{Y}$ (set of class labels). $Y(x)$ is a random variable modeling output for input x. ζ represents a particular classifier with its GE denoted by

$GE(\zeta)$. $\mathcal{Z}(N)$ denotes a set of classifiers obtained by application of a classification algorithm to different samples of size N.

3.1 Technical Framework

The basic idea in the generic characterization of the moments of GE, is to define a class of classifiers induced by a classification algorithm and an i.i.d. sample of a particular size from an underlying distribution. Each classifier in this class and its GE act as random variables, since the process of obtaining the sample is randomized. Since $GE(\zeta)$ is a random variable, it has a distribution. Quite often though, characterizing a finite subset of moments turns out to be a more viable option than characterizing the entire distribution. Based on these facts, we revisit the expressions for the first two moments around zero of the GE of a classifier,

$$E_{\mathcal{Z}(N)}\left[GE(\zeta)\right] = \sum_{x\in\mathcal{X}} P\left[X=x\right] \sum_{y\in\mathcal{Y}} P_{\mathcal{Z}(N)}\left[\zeta(x)=y\right] P\left[Y(x)\neq y\right],$$

$$E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[GE(\zeta)GE(\zeta')\right] = \sum_{x\in\mathcal{X}}\sum_{x'\in\mathcal{X}} P\left[X=x\right] P\left[X=x'\right] \cdot \sum_{y\in\mathcal{Y}}\sum_{y'\in\mathcal{Y}} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[\zeta(x)=y \wedge \zeta'(x')=y'\right] \cdot P\left[Y(x)\neq y\right] P\left[Y(x')\neq y'\right]$$

From the above equations we observe that for the first moment we have to characterize the behavior of the classifier on each input separately while for the second moment we need to observe its behavior on pairs of inputs. In particular, to derive expressions for the moments of any classification algorithm we need to characterize $P_{\mathcal{Z}(N)}\left[\zeta(x)=y\right]$ for the first moment and $P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=y\wedge\zeta'(x')=y']$ for the second moment.[2] The values for the other terms denote the error of the classifier for the first moment and errors of two classifiers for the second moment which are obtained directly from the underlying joint distribution. For example, if we have data with a class prior p for class 1 and *1-p* for class 2. Then the error of a classifier classifying data into class 1 is *1-p* and the error of a classifier classifying data into class 2 is given by p. We now focus our attention on relating the above two probabilities, to probabilities that can be computed using the joint distribution and the classification model viz. Decision Trees.

In the subsections that follow we assume the following setup. We consider the dimensionality of the input space to be d. $A_1, A_2, ..., A_d$ are the corresponding discrete attributes or continuous attributes with predetermined split points. $a_1, a_2, ..., a_d$ are the number of attribute values/the number of splits of the attributes $A_1, A_2, ..., A_d$ respectively. m_{ij} is the i^{th} attribute value/split of the j^{th} attribute, where $i \leq a_j$ and $j \leq d$. Let $C_1, C_2, ..., C_k$ be the class labels representing k classes and N the sample size.

2. These probabilities and $P\left[Y(x)\neq y\right]$ are conditioned on x. We omit explicitly writing the conditional since it improves readability and is obvious from the context.

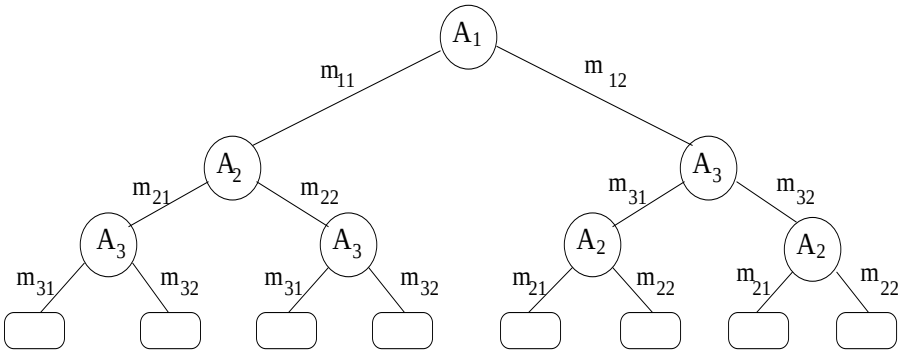


Figure 1: The all attribute tree with 3 attributes $A_1,\ A_2,\ A_3$, each having 2 values.

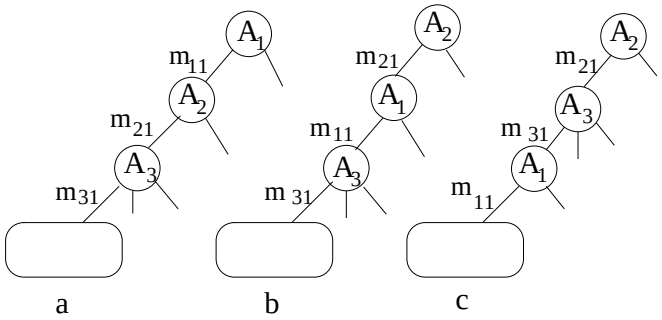


Figure 2: Given 3 attributes $A_1,\ A_2,\ A_3$, the path $m_{11}m_{21}m_{31}$ is formed irrespective of the ordering of the attributes. Three such permutations are shown in the above figure.

3.2 All Attribute Decision Trees (ATT)

Let us consider a decision tree algorithm whose only stopping criterion is that no attributes remain when building any part of the tree. In other words, every path in the tree from root to leaf has all the attributes. An example of such a tree is shown in Figure 1. It can be seen that irrespective of the split attribute selection method (e.g., information gain, gini gain, randomized selection, etc.) the above stopping criteria yields trees with the same leaf nodes. Thus although a particular path in one tree has an ordering of attributes that might be different from a corresponding path in other trees, the leaf nodes will represent the same region in space or the same set of datapoints. This is seen in Figure 2. Moreover, since predictions are made using data in the leaf nodes, any deterministic way of prediction would lead to these trees resulting in the same classifier for a given sample and thus having the same GE. Usually, prediction in the leaves is performed by choosing the most numerous class as the class label for the corresponding datapoint. With this we arrive at the expressions for computing the aforementioned probabilities,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)}\left[\zeta(x)=C_i\right]= \\
&\quad P_{\mathcal{Z}(N)}[ct(m_{p1}m_{q2}...m_{rd}C_i)>ct(m_{p1}m_{q2}...m_{rd}C_j), \\
&\qquad \forall j\neq i,\ i,j\in[1,...,k]]
\end{aligned}
$$

where $x = m_{p1}m_{q2}...m_{rd}$ denotes a datapoint which is also a path from root to leaf in the tree. We refer to this path as a cell sometimes since it represents a rectangular region in a d dimensional space. $ct(m_{p1}m_{q2}...m_{rd}C_i)$ is the count of the datapoints in the cell $m_{p1}m_{q2}...m_{rd}C_i$. Henceforth, when using the word "path" we will strictly imply path from root to leaf. By computing the above probability $\forall\ i$ and $\forall\ x$ we can compute the first moment of the GE for this classification algorithm.

Similarly, for the second moment we compute cumulative joint probabilities of the following form:

$$\begin{aligned} &P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[\zeta(x)=C_i \wedge \zeta'(x')=C_v\right] = \\ &\quad P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(m_{p1}...m_{rd}C_i) > ct(m_{p1}...m_{rd}C_j), \\ &\qquad ct(m_{f1}...m_{hd}C_v) > ct(m_{f1}...m_{hd}C_w), \\ &\qquad \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]] \end{aligned}$$

where the terms have similar connotation as before. These probabilities can be computed exactly or by using fast approximation techniques proposed in Dhurandhar and Dobra (2009).

3.3 Decision Trees with Non-trivial Stopping Criteria

We just considered decision trees which are grown until all attributes are exhausted. In real life though we seldom build such trees. The main reasons for this could be any of the following: we wish to build small decision trees to save space; certain path counts (i.e., number of datapoints in the leaves) are extremely low and hence we want to avoid splitting further, as the predictions can get arbitrarily bad; we have split on a certain subset of attributes and all the datapoints in that path belong to the same class (purity based criteria); we want to grow trees to a fixed height (or depth). These stopping measures would lead to paths in the tree that contain a subset of the entire set of attributes. Thus from a classification point of view we cannot simply compare the counts in two cells as we did previously. The reason for this being that the corresponding path may not be present in the tree. Hence, we need to check that the path exists and then compare cell counts. Given the classification algorithm, since the $P_{\mathcal{Z}(N)}\left[\zeta(x)=C_i\right]$ is the probability of all possible ways in which an input x can be classified into class C_i for a decision tree it equates to finding the following kind of probability for the first moment,

$$\begin{aligned} &P_{\mathcal{Z}(N)}\left[\zeta(x)=C_i\right] = \\ &\quad \sum_p P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), path_p exists, \\ &\qquad \forall j \neq i,\ i,j \in [1,...,k]] \end{aligned} \tag{1}$$

where p indexes all allowed paths by the tree algorithm in classifying input x. After the summation, the right hand side term above is the probability that the cell $path_pC_i$ has the greatest count, with the path "$path_p$" being present in the tree. This will become clearer when we discuss different stopping criteria. Notice that the characterization for the ATT is just a special case of this more generic characterization.

The probability that we need to find for the second moment is,

$$\begin{aligned} &P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[\zeta(x)=C_i \wedge \zeta'(x')=C_v\right] = \\ &\sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), path_pexists, \\ &\qquad ct(path_qC_v) > ct(path_qC_w), path_qexists, \\ &\qquad \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]] \end{aligned} \tag{2}$$

where p and q index all allowed paths by the tree algorithm in classifying input x and x' respectively. The above two equations are generic in analyzing any decision tree algorithm which classifies inputs into the most numerous class in the corresponding leaf. It is not difficult to generalize it further when the decision in leaves is some other measure than majority. In that case we would just include that measure in the probability in place of the inequality.

3.3.1 CHARACTERIZING *Path Exists* FOR THREE STOPPING CRITERIA

It follows from above that to compute the moments of the GE for a decision tree algorithm we need to characterize conditions under which particular paths are present. This characterization depends on the stopping criteria and split attribute selection method in a decision tree algorithm. We now look at three popular stopping criteria, namely a) Fixed height based, b) Purity (i.e., entropy 0 or gini index 0 etc.) based and c) Scarcity (i.e., too few datapoints) based. We consider conditions under which certain paths are present for each stopping criteria. Similar conditions can be enumerated for any reasonable stopping criteria. We then choose a split attribute selection method, thereby fully characterizing the above two probabilities and hence the moments.

1. **Fixed Height:** This stopping criteria is basically that every path in the tree should be of length exactly h, where $h \in [1,...,d]$. If $h = 1$ we classify based on just one attribute. If $h = d$ then we have the all attribute tree.
   In general, a path $m_{i1}m_{j2}...m_{lh}$ is present in the tree iff the attributes $A_1,\ A_2,...,\ A_h$ are chosen in any order to form the path for a tree construction during the split attribute selection phase. Thus, for any path of length h to be present we bi-conditionally imply that the corresponding attributes are chosen.

2. **Purity:** This stopping criteria implies that we stop growing the tree from a particular split of a particular attribute if all datapoints lying in that split belong to the same class. We call such a path pure else we call it impure. In this scenario, we could have paths of length 1 to d depending on when we encounter purity (assuming all datapoints don't lie in 1 class). Thus, we have the following two separate checks for paths of length d and less than d respectively.

   a) Path $m_{i1}m_{j2}...m_{ld}$ present iff the path $m_{i1}m_{j2}...m_{l(d-1)}$ is impure and attributes $A_1,\ A_2,...,\ A_{d-1}$ are chosen above A_d, or $m_{i1}m_{j2}...m_{s(d-2)}m_{ld}$ is impure and attributes $A_1,\ A_2,...,\ A_{d-2},\ A_d$ are chosen above A_{d-1}, or ... or $m_{j2}...m_{ld}$ is impure and attributes $A_2,...,\ A_d$ are chosen above A_1.
   This means that if a certain set of $d-1$ attributes are present in a path in the tree then we split on the d^{th} attribute iff the current path is not pure, finally resulting in a path of length d.

   b) Path $m_{i1}m_{j2}...m_{lh}$ present where $h < d$ iff the path $m_{i1}m_{j2}...m_{lh}$ is pure and attributes $A_1,\ A_2,...,\ A_{h-1}$ are chosen above A_h and $m_{i1}m_{j2}...m_{l(h-1)}$ is impure or the path $m_{i1}m_{j2}...m_{lh}$

is pure and attributes A_1, A_2,..., A_{h-2}, A_h are chosen above A_{h-1} and $m_{i1}m_{j2}...m_{l(h-2)}m_{lh}$ is impure or ... or the path $m_{j2}...m_{lh}$ is pure and attributes A_2,..., A_h are chosen above A_1 and $m_{j2}...m_{lh}$ is impure.
This means that if a certain set of $h-1$ attributes are present in a path in the tree then we split on some h^{th} attribute iff the current path is not pure and the resulting path is pure.

The above conditions suffice for "path present" since the purity property is anti-monotone and the impurity property is monotone.

3. **Scarcity:** This stopping criteria implies that we stop growing the tree from a particular split of a certain attribute if its count is less than or equal to some pre-specified pruning bound. Let us denote this number by pb. As before, we have the following two separate checks for paths of length d and less than d respectively.

   a) Path $m_{i1}m_{j2}...m_{ld}$ present iff the attributes $A_1,...,A_{d-1}$ are chosen above A_d and $ct(m_{i1}m_{j2}...m_{l(d-1)}) > pb$ or the attributes $A_1,...,A_{d-2},A_d$ are chosen above A_{d-1} and $ct(m_{i1}m_{j2}...m_{l(d-2)}m_{nd}) > pb$ or ... or the attributes $A_2,...,A_d$ are chosen above A_1 and $ct(m_{i2}m_{j3}...m_{ld}) > pb$.

   b) Path $m_{i1}m_{j2}...m_{lh}$ present where $h < d$ iff the attributes $A_1,...,A_{h-1}$ are chosen above A_h and $ct(m_{i1}m_{j2}...m_{l(h-1)}) > pb$ and $ct(m_{i1}m_{j2}...m_{lh}) \leq pb$ or the attributes $A_1,...,A_{h-2},A_h$ are chosen above A_{h-1} and $ct(m_{i1}m_{j2}...m_{l(h-2)}m_{nh}) > pb$ and $ct(m_{i1}m_{j2}...m_{nh}) \leq pb$ or ... or the attributes $A_2,...,A_h$ are chosen above A_1 and $ct(m_{i2}m_{j3}...m_{lh}) > pb$ and $ct(m_{i1}m_{j2}...m_{lh}) \leq pb$.
   This means that we stop growing the tree under a node once we find that the next chosen attribute produces a path with occupancy $\leq pb$.

   The above conditions suffice for "path present" since the occupancy property is monotone.

We observe from the above checks that we have two types of conditions that need to be evaluated for a path being present namely, i) those that depend on the sample viz. $m_{i1}m_{j2}...m_{l(d-1)}$ is impure or $ct(m_{i1}m_{j2}...m_{lh}) > pb$ and ii) those that depend split attribute selection method viz. A_1, A_2,..., A_h are chosen. The former depends on the data distribution which we have specified to be a multinomial. The latter we discuss in the next subsection. Note that checks for a combination of the above stopping criteria can be obtained by appropriately combining the individual checks.

3.4 Split Attribute Selection

In decision tree construction algorithms, at each iteration we have to decide the attribute variable on which the data should be split. Numerous measures have been developed (Hall and Holmes, 2003). Some of the most popular ones aim to increase the purity of a set of datapoints that lie in the region formed by that split. The purer the region, the better the prediction and lower the error of the classifier. Measures such as, i) Information Gain (IG) (Quinlan, 1986), ii) Gini Gain (GG) (Breiman et al., 1984), iii) Gain Ratio (GR) (Quinlan, 1986), iv) Chi-square test (CS) (Shao, 2003) etc. aim at realizing this intuition. Other measures using Principal Component Analysis (Smith, 2002), Correlation-based measures (Hall, 1998) have also been developed. Another interesting yet non-intuitive measure in terms of its utility is the Random attribute selection measure. According to this measure we randomly choose the split attribute from available set. The decision tree that this algorithm produces is called a Random decision tree (RDT). Surprisingly enough, a collection of RDTs quite often outperform their seemingly more powerful counterparts (Liu et al., 2005). In this

paper we study this interesting variant. We do this by first presenting a probabilistic characterization in selecting a particular attribute/set of attributes, followed by simulation studies. Characterizations for the other measures can be developed in similar vein by focusing on the working of each measure. As an example, for the deterministic purity based measures mentioned above the split attribute selection is just a function of the sample and thus by appropriately conditioning on the sample we can find the relevant probabilities and hence the moments.

Before presenting the expression for the probability of selecting a split attribute/attributes in constructing a RDT we extend the results in Dhurandhar and Dobra (2009) where relationships were drawn between the moments of HE, CE, LE (just a special case of cross validation) and GE, to be applicable to randomized classification algorithms. The random process is assumed to be independent of the sampling process. This result is required since the results in Dhurandhar and Dobra (2009) are applicable to deterministic classification algorithms and we would be analyzing RDT's. With this we have the following lemma.

Lemma 1 *Let D and T be independent discrete random variables, with some distribution defined on each of them. Let $\mathcal{D}$ and $\mathcal{T}$ denote the domains of the random variables. Let $f(d,t)$ and $g(d,t)$ be two functions such that $\forall t \in \mathcal{T}\ E_{\mathcal{D}}[f(d,t)] = E_{\mathcal{D}}[g(d,t)]$ and $d \in \mathcal{D}$. Then, $E_{\mathcal{T}\times\mathcal{D}}[f(d,t)] = E_{\mathcal{T}\times\mathcal{D}}[g(d,t)]$*

Proof

$$\begin{aligned} E_{\mathcal{T}\times\mathcal{D}}[f(d,t)] &= \sum_{t\in\mathcal{T}}\sum_{d\in\mathcal{D}} f(d,t)P[T=t,D=d] \\ &= \sum_{t\in\mathcal{T}}\sum_{d\in\mathcal{D}} f(d,t)P[D=d]P[T=t] \\ &= \sum_{t\in\mathcal{T}} E_{\mathcal{D}}[g(d,t)]P[T=t] \\ &= E_{\mathcal{T}\times\mathcal{D}}[g(d,t)]. \end{aligned}$$

■

The result is valid even when D and T are continuous, but considering the scope of this paper we are mainly interested in the discrete case. This result implies that all the relationships and expressions in Dhurandhar and Dobra (2009) hold with an extra expectation over the $t's$, for randomized classification algorithms where the random process is independent of the sampling process.

3.5 Random Decision Trees

In this subsection we explain the randomized process used for split attribute selection and provide the expression for the probability of choosing an attribute/a set of attributes. The attribute selection method we use is as follows. We assume a uniform probability distribution in selecting the attribute variables, that is, attributes which have already not been chosen in a particular branch, have an equal chance of being chosen for the next level. The random process involved in attribute selection is independent of the sample and hence the lemma 1 applies. We now give the expression for the probability of selecting a subset of attributes from the given set for a path. This expression is required in the computation of the above mentioned probabilities used in computing the moments.

For the first moment we need to find the following probability. Given d attributes A_1, A_2,..., A_d the probability of choosing a set of h attributes where $h \in \{1,2,...,d\}$ is,

$$P[h \text{ attributes chosen}] = \frac{1}{\left(\begin{array}{c} d \\ h \end{array} \right)}$$

since choosing without replacement is equivalent to simultaneously choosing a subset of attributes from the given set.

For the second moment when the trees are different (required in the finding of variance of CE since, the training sets in the various runs in cross validation are different, that is, for finding $E_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[GE(\zeta)GE(\zeta')]$), the probability of choosing l_1 attributes for path in one tree and l_2 attributes for path in another tree where $l_1, l_2 \leq d$ is given by,

$$P[l_1 \text{ attribute path in tree } 1,\ l_2 \text{ attribute path in tree } 2] = \frac{1}{\left(\begin{array}{c} d \\ l_1 \end{array} \right)\left(\begin{array}{c} d \\ l_2 \end{array} \right)}$$

since the process of choosing one set of attributes for a path in one tree is independent of the process of choosing another set of attributes for a path in a different tree.

For the second moment when the tree is the same (required in the finding of variance of GE and HE, that is, for finding $E_{\mathcal{Z}(N)}\left[GE(\zeta)^2\right]$), the probability of choosing two sets of attributes such that the two distinct paths resulting from them co-exist in a single tree is given by the following. Assume we have d attributes A_1, A_2,..., A_d. Let the lengths of the two paths (or cardinality of the two sets) be l_1 and l_2 respectively, where $l_1, l_2 \leq d$. Without loss of generality assume $l_1 \leq l_2$. Let p be the number of attributes common to both paths. Notice that $p \geq 1$ is one of the necessary conditions for the two paths to co-exist. Let $v \leq p$ be those attributes among the total p that have same values for both paths. Thus $p - v$ attributes are common to both paths but have different values. At one of these attributes in a given tree the two paths will bifurcate. The probability that the two paths co-exist given our randomized attribute selection method is computed by finding out all possible ways in which the two paths can co-exist in a tree and then multiplying the number of each kind of way by the probability of having that way. A detailed proof is given in the Appendix A. The expression for the probability based on the attribute selection method is,

$$P[l_1 \text{ and } l_2 \text{ length paths co} - \text{exist}] =$$
$$\sum_{i=0}^{v} vPr_i (l_1 - i - 1)!(l_2 - i - 1)!(p - v) prob_i$$

where $vPr_i = \frac{v!}{(v-i)!}$ denotes permutation and $prob_i = \frac{1}{d(d-1)...(d-i)(d-i-1)^2...(d-l_1+1)^2(d-l_1)...(d-l_2+1)}$ is the probability of the i^{th} possible way. For fixed height trees of height h, $(l_1 - i - 1)!(l_2 - i - 1)!$ becomes $(h - i - 1)!^2$ and $prob_i = \frac{1}{d(d-1)...(d-i)(d-i-1)^2...(d-h+1)^2}$.

3.6 Putting Things Together

We now have all the ingredients that are required for the computation of the moments of GE. In this subsection we combine the results derived in the previous subsections to obtain expressions for

$P_{\mathcal{Z}(N)}[\zeta(x)=C_i]$ and $P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[\zeta(x)=C_i \wedge \zeta'(x')=C_v]$ which are vital in the computation of the moments.

Let *s.c.c.s.* be an abbreviation for stopping criteria conditions that are sample dependent. Conversely, *s.c.c.i.* be an abbreviation for stopping criteria conditions that are dependent on the attribute selection method. We now provide expressions for the above probabilities categorized by the 3 stopping criteria.

3.6.1 FIXED HEIGHT

The conditions for "path exists" for fixed height trees depend only on the attribute selection method as seen in Section 3.3.1. Hence the probability used in finding the first moment is given by,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)}[\zeta(x)=C_i] \\
&= \sum_p P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), path_p exists,\ \forall j \neq i,\ i,j \in [1,...,k]] \\
&= \sum_p P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), s.c.c.i.,\ \forall j \neq i,\ i,j \in [1,...,k]] \\
&= \sum_p P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j),\ \forall j \neq i,\ i,j \in [1,...,k]] P_{\mathcal{Z}(N)}[s.c.c.i.] \\
&= \sum_p \frac{P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j),\ \forall j \neq i,\ i,j \in [1,...,k]]}{\binom{d}{h}}
\end{aligned}
$$

where h is the length of the paths or the height of the tree. The probability in the last step of the above derivation can be computed from the underlying joint distribution. The probability for the second moment when the trees are different is given by,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[\zeta(x)=C_i \wedge \zeta'(x')=C_v\right] \\
&= \sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), path_p exists, ct(path_qC_v) > ct(path_qC_w), \\
&\quad path_q exists, \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]] \\
&= \sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), ct(path_qC_v) > ct(path_qC_w), \forall j \neq i, \\
&\quad \forall w \neq v,\ i,j,v,w \in [1,...,k]] \cdot P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[s.c.c.i.] \\
&= \frac{1}{\binom{d}{h}^2} \Big(\sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), ct(path_qC_v) > ct(path_qC_w), \\
&\qquad \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]]\Big).
\end{aligned}
$$

The probability for the second moment when the trees are the same is given by,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)}\left[\zeta(x)=C_i \wedge \zeta(x')=C_v\right] \\
&=\sum_{p,q} P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), path_p exists, ct(path_qC_v)>ct(path_qC_w), \\
&\quad path_q exists, \forall j\neq i,\ \forall w\neq v,\ i,j,v,w\in[1,...,k]] \\
&=\sum_{p,q} P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), ct(path_qC_v)>ct(path_qC_w), \forall j\neq i,\ \forall w\neq v,\ i,j, \\
&\quad v,w\in[1,...,k]]\cdot P_{\mathcal{Z}(N)}[s.c.c.i.] \\
&=\sum_{p,q}\sum_{t=0}^{b} bPr_t(h-t-1)!^2(r-v)prob_t P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), \\
&\qquad ct(path_qC_v)>ct(path_qC_w), \forall j\neq i,\ \forall w\neq v,\ i,j,v,w\in[1,...,k]]
\end{aligned}
$$

where r is the number of attributes that are common in the 2 paths, b is the number of attributes that have the same value in the 2 paths, h is the length of the paths and $prob_t = \frac{1}{d(d-1)...(d-t)(d-t-1)^2...(d-h+1)^2}$. As before, the probability comparing counts can be computed from the underlying joint distribution.

3.6.2 PURITY AND SCARCITY

The conditions for "path exists" in the case of purity and scarcity depend on both the sample and the attribute selection method as can be seen in 3.3.1. The probability used in finding the first moment is given by,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)}\left[\zeta(x)=C_i\right] \\
&=\sum_{p} P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), path_p exists,\ \forall j\neq i,\ i,j\in[1,...,k]] \\
&=\sum_{p} P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), s.c.c.i, s.c.c.s.,\ \forall j\neq i,\ i,j\in[1,...,k]] \\
&=\sum_{p} P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), s.c.c.s.,\ \forall j\neq i,\ i,j\in[1,...,k]]P_{\mathcal{Z}(N)}[s.c.c.i.] \\
&=\sum_{p} \frac{P_{\mathcal{Z}(N)}[ct(path_pC_i)>ct(path_pC_j), s.c.c.s.,\ \forall j\neq i,\ i,j\in[1,...,k]]}{dC_{h_p-1}(d-h_p+1)}
\end{aligned}
$$

where h_p is the length of the path indexed by p. The joint probability of comparing counts and $s.c.c.s.$ can be computed from the underlying joint distribution. The probability for the second moment when the trees are different is given by,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}\left[\zeta(x)=C_i \wedge \zeta'(x')=C_v\right] \\
&= \sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), path_pexists, ct(path_qC_v) > ct(path_qC_w), \\
&\quad path_qexists, \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]] \\
&= \sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), ct(path_qC_v) > ct(path_qC_w), s.c.c.s., \forall j \neq i, \\
&\quad \forall w \neq v,\ i,j,v,w \in [1,...,k]] \cdot P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[s.c.c.i.] \\
&= \frac{1}{dC_{h_p-1}dC_{h_q-1}(d-h_p+1)(d-h_q+1)}(\sum_{p,q} P_{\mathcal{Z}(N)\times\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), \\
&\quad ct(path_qC_v) > ct(path_qC_w), s.c.c.s., \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]])
\end{aligned}
$$

where h_p and h_q are the lengths of the paths indexed by p and q. The probability for the second moment when the trees are the same is given by,

$$
\begin{aligned}
&P_{\mathcal{Z}(N)}\left[\zeta(x)=C_i \wedge \zeta(x')=C_v\right] \\
&= \sum_{p,q} P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), path_pexists, ct(path_qC_v) > ct(path_qC_w), path_qexists, \\
&\quad \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]] \\
&= \sum_{p,q} P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), ct(path_qC_v) > ct(path_qC_w), s.c.c.s., \forall j \neq i,\ \forall w \neq v, \\
&\quad i,j,v,w \in [1,...,k]] P_{\mathcal{Z}(N)}[s.c.c.i.] \\
&= \sum_{p,q}\sum_{t=0}^{b} \frac{bPr_t(h_p-t-2)!(h_q-t-2)!(r-v)prob_t}{(d-h_p+1)(d-h_q+1)} P_{\mathcal{Z}(N)}[ct(path_pC_i) > ct(path_pC_j), \\
&\quad ct(path_qC_v) > ct(path_qC_w), s.c.c.s., \forall j \neq i,\ \forall w \neq v,\ i,j,v,w \in [1,...,k]]
\end{aligned}
$$

where r is the number of attributes that are common in the 2 paths sparing the attributes chosen as leaves, b is the number of attributes that have the same value, h_p and h_q are the lengths of the 2 paths and without loss of generality assuming $h_p \leq h_q$ $prob_t = \frac{1}{d(d-1)...(d-t)(d-t-1)^2...(d-h_p)^2(d-h_p-1)...(d-h_q)}$. As before, the probability of comparing counts and s.c.c.s. can be computed from the underlying joint distribution.

Using the expressions for the above probabilities the moments of GE can be computed. In next section we perform experiments on synthetic as well as distributions built on real data to portray the efficacy of the derived expressions.

4. Experiments

To exactly compute the probabilities for each path the time complexity for fixed height trees is $O(N^2)$ and for purity and scarcity based trees it is $O(N^3)$. Hence, computing exactly the probabilities and consequently the moments is practical for small values of N. For larger values of N, we propose computing the individual probabilities using Monte Carlo. In the empirical studies we report, we initially set N to small value and compute the error (i.e., expected value + standard deviation) exactly, using the derived expressions (which is thus the golden standard) and compare it

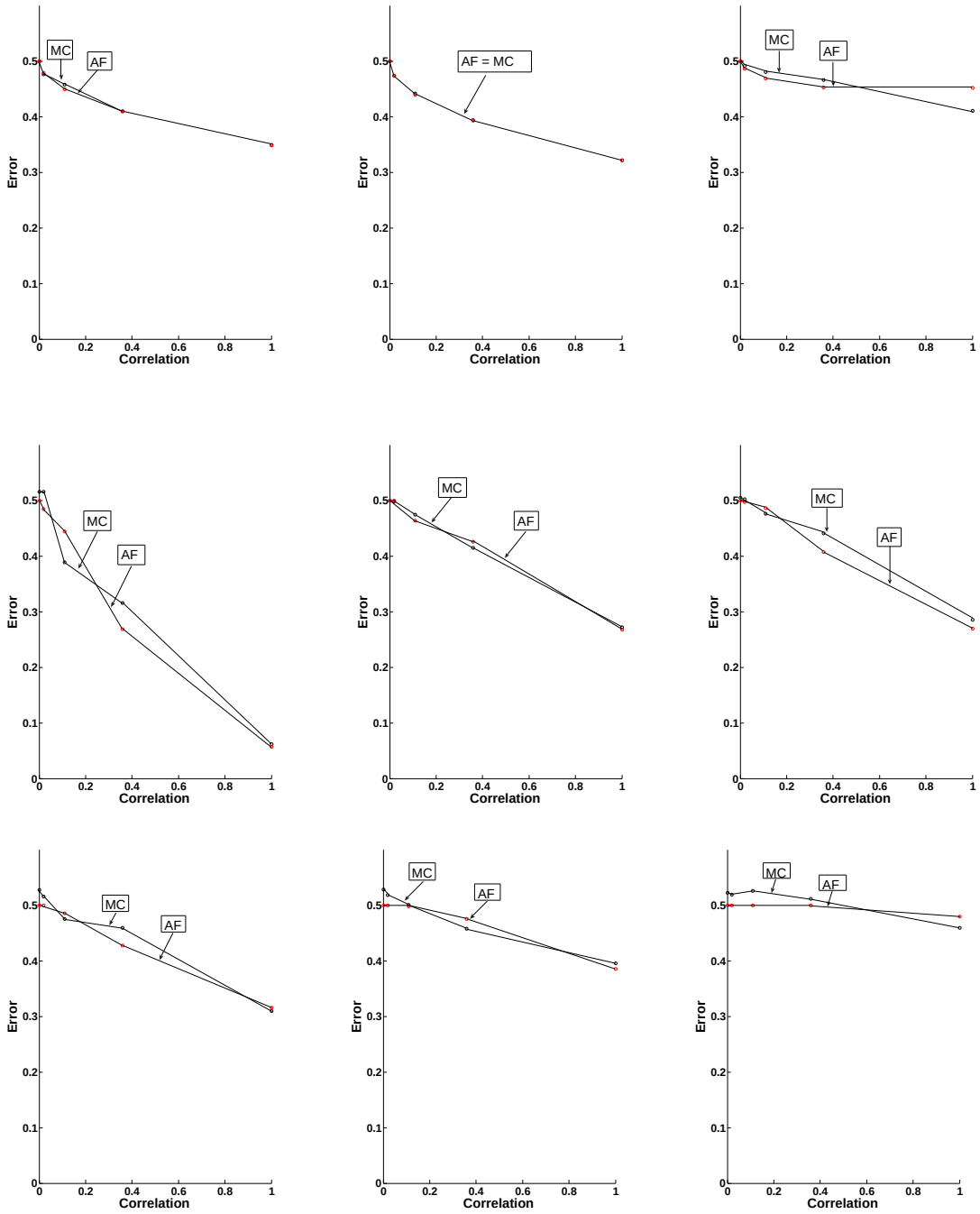


Figure 3: Errors of Fixed height trees (top row figures), Purity trees (center row figures) and Scarcity trees (bottom row figures) with $N = 100$ are shown. The leftmost figures are for $d = 5$ and binary splits, the center figures are for $d = 5$ and ternary splits and the rightmost figures are for $d = 8$ and binary splits. $h = 3$ for Fixed height trees and $pb = \frac{N}{10}$ for Scarcity based trees.

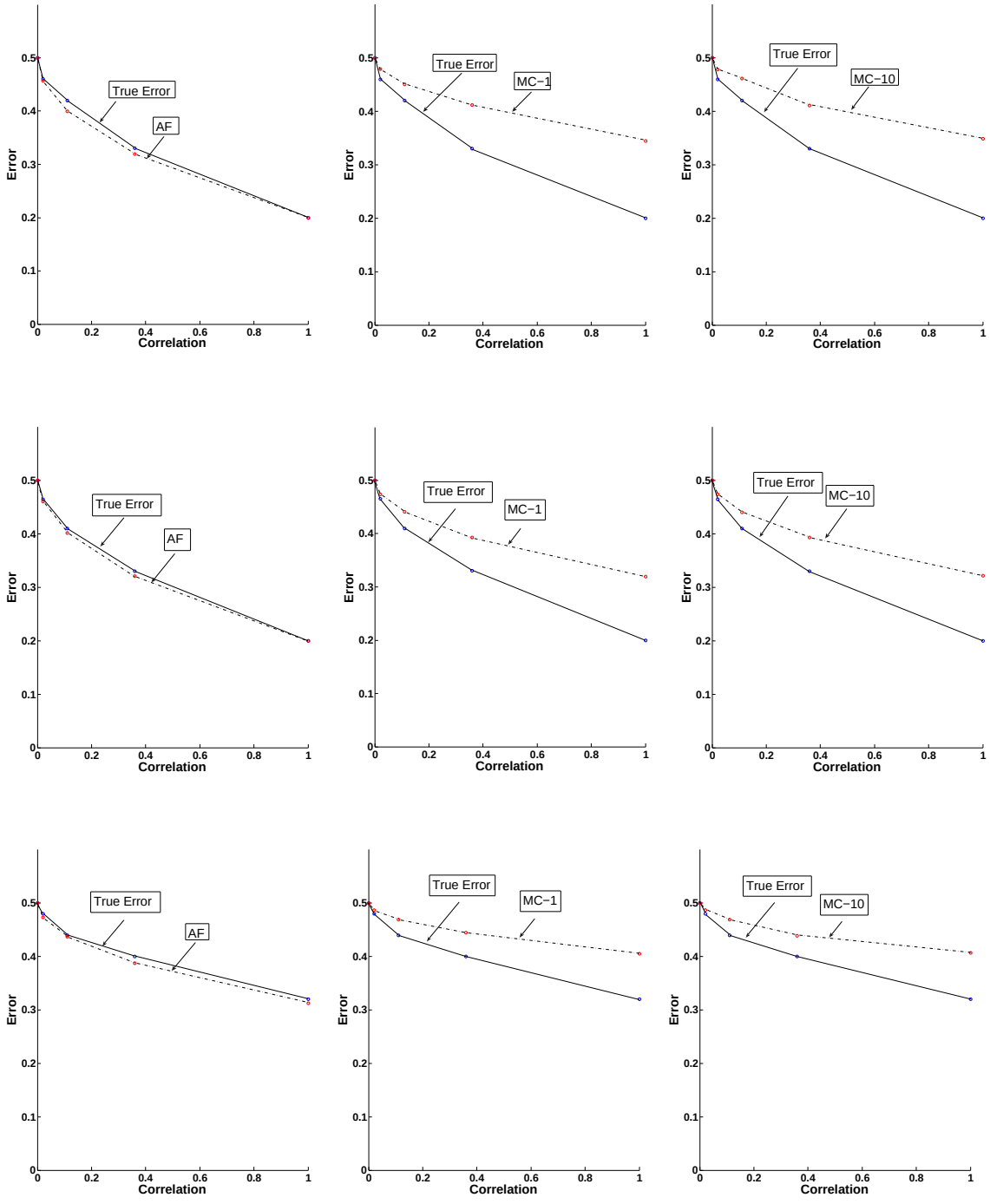


Figure 4: Errors of Fixed height trees with $N = 10000$ and $h = 3$ are shown. In the top row $d = 5$ and splits are binary, in the center row $d = 5$ and splits are ternary and in the last row $d = 8$ and splits are binary.

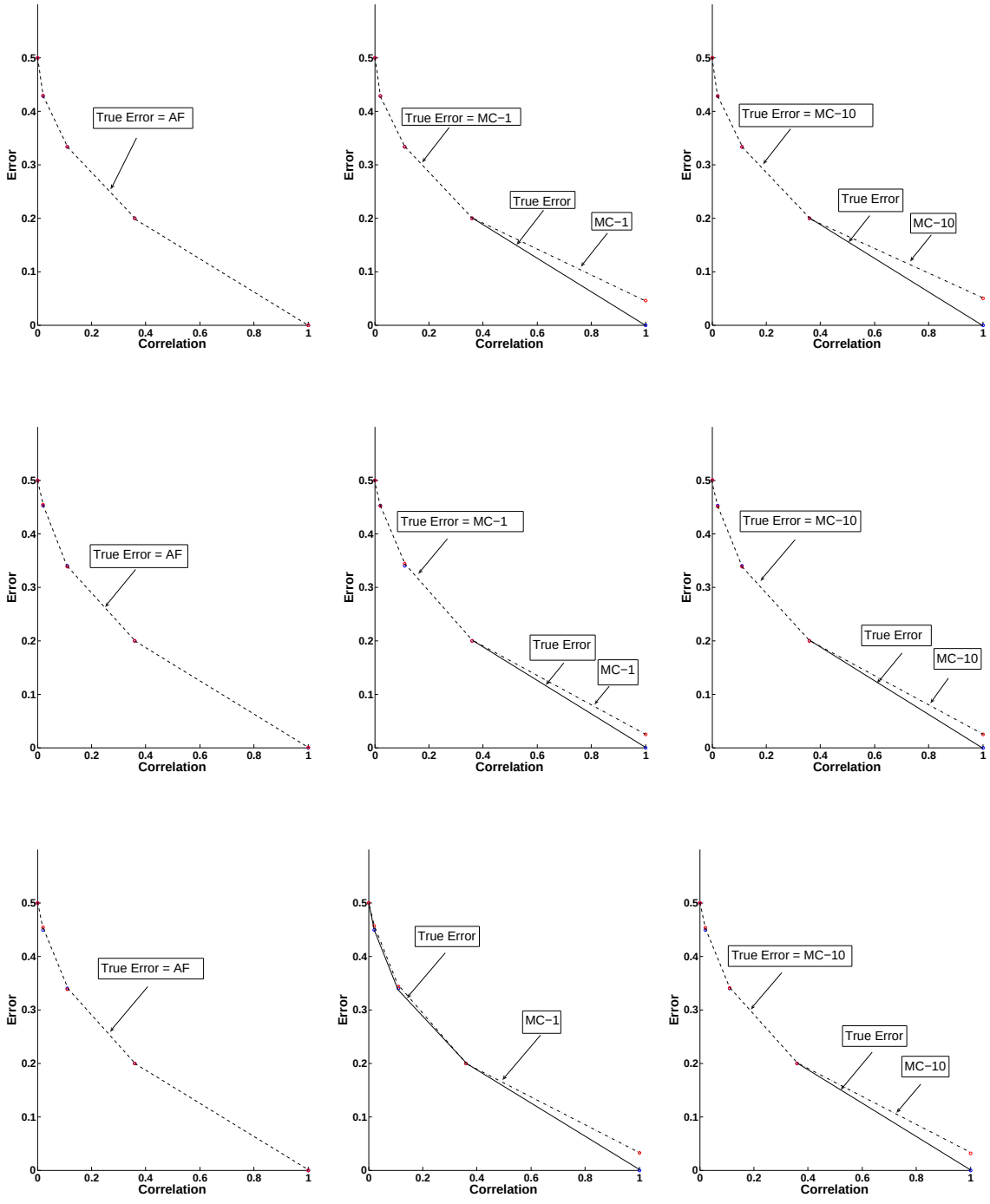


Figure 5: Errors of Purity trees with $N = 10000$ are shown. In the top row $d = 5$ and splits are binary, in the center row $d = 5$ and splits are ternary and in the last row $d = 8$ and splits are binary.

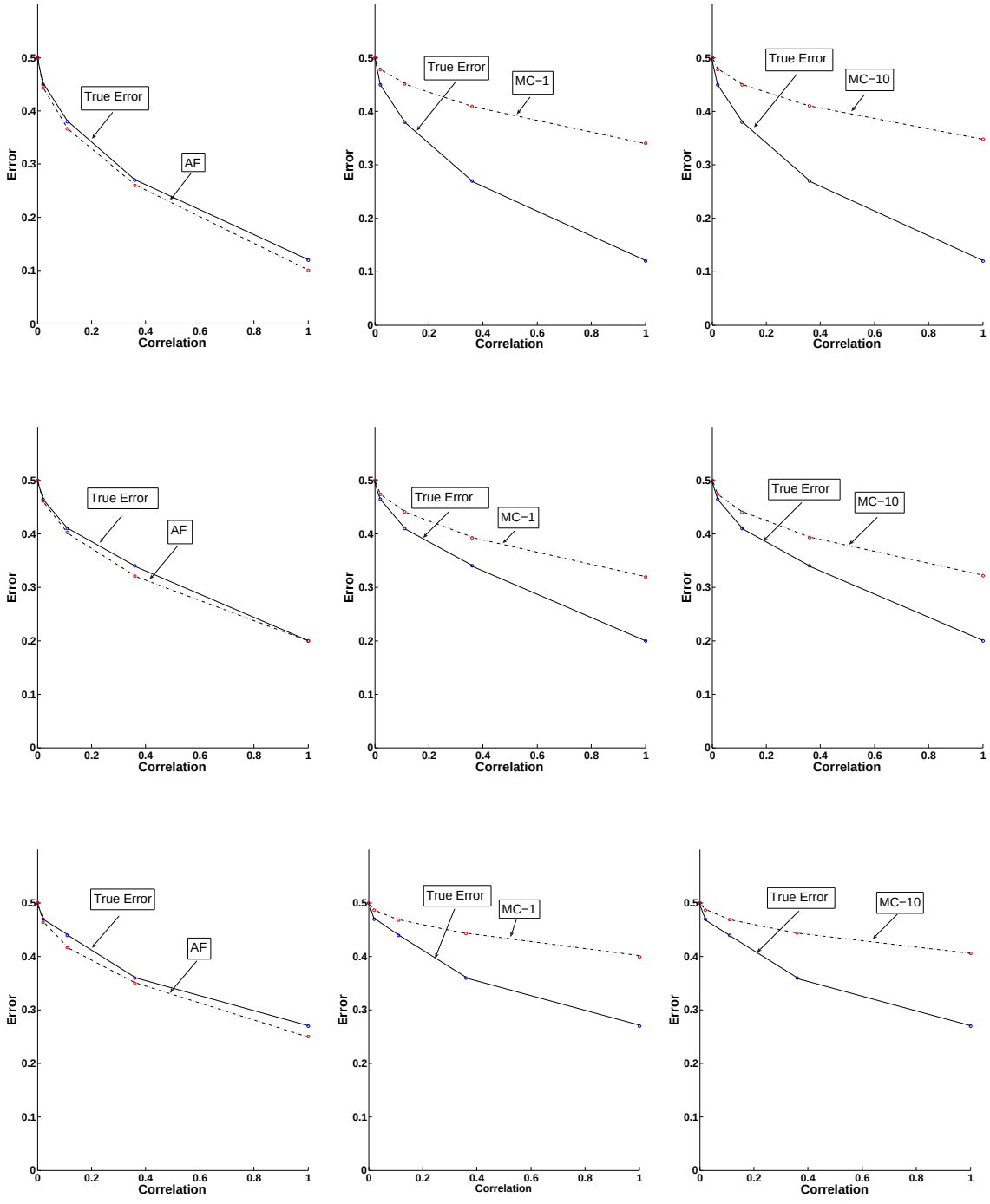


Figure 6: Errors of Scarcity trees with $N = 10000$ and $pb = \frac{N}{10}$ are shown. In the top row $d = 5$ and splits are binary, in the center row $d = 5$ and splits are ternary and in the last row $d = 8$ and splits are binary.

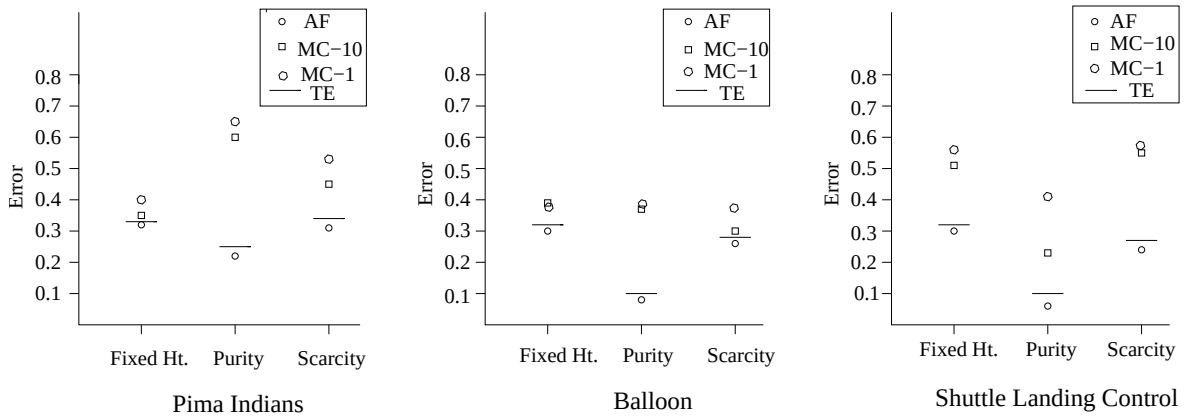


Figure 7: Comparison between AF and MC on three UCI data sets for trees prunned based on fixed height ($h = 3$), purity and scarcity ($pb = \frac{N}{10}$).

Stopping Criteria	Split	$\rho = 1$	$\rho = 0.36$	$\rho = 0.11$	$\rho = 0.02$	$\rho = 0$
Fixed Height						
$N = 100$, $d = 5$, $h = 3$	binary	29.67	1.49	0.56	0.34	0.51
$N = 100$, $d = 5$, $h = 3$	ternary	277.37	20.49	10.77	7.7	9.23
$N = 100$, $d = 8$, $h = 3$	binary	152.21	3.89	2.78	1.33	1.57
$N = 10000$, $d = 5$, $h = 3$	binary	41.89	2.99	1.25	0.78	0.71
$N = 10000$, $d = 5$, $h = 3$	ternary	575.15	30.9	15.71	11.87	10.8
$N = 10000$, $d = 8$, $h = 3$	binary	1813.86	7.21	3.86	2.56	2.3
Purity						
$N = 100$, $d = 5$	binary	39.67	1154.1	5216.75	10783.19	13750.28
$N = 100$, $d = 5$	ternary	160.59	181.21	180.5	3281.83	6884.52
$N = 100$, $d = 8$	binary	2.8	1.9	1035.68	1211.7	1249.32
$N = 10000$, $d = 5$	binary	40.54	2897.3	11499.57	65581.6	422011.93
$N = 10000$, $d = 5$	ternary	1386.01	163245.31	675867.31	2662617.25	5781240
$N = 10000$, $d = 8$	binary	221.98	178913.85	712081.12	3113403.25	6885975
Scarcity						
$N = 100$, $d = 5$	binary	17.17	17.59	17.5	17.2	17.08
$N = 100$, $d = 5$	ternary	34.10	33.55	32.88	32.18	31.52
$N = 100$, $d = 8$	binary	34.42	33.86	33.28	32.59	31.89
$N = 10000$, $d = 5$	binary	13.04	12.18	11.26	10.32	9.38
$N = 10000$, $d = 5$	ternary	61.01	60.34	59.51	58.64	57.76
$N = 10000$, $d = 8$	binary	2643.21	2642.56	2641.75	2640.89	2640.04

Table 1: The above table shows the upper bounds on $E_{\mathcal{Z}(N)}[GE(\zeta)]$ for different levels of correlation (ρ) between the attributes and class labels obtained using Breiman's formula.

Stopping Criteria	Pima Indians	Balloon	Shuttle Landing Control
Fixed Height	151.58	51.84	1.91
Purity	98.97	50.56	2.74
Scarcity	180.93	41.67	2.32

Table 2: The above table shows the upper bounds on $E_{\mathcal{Z}(N)}[GE(\zeta)]$ for 3 UCI data sets obtained using Breiman's formula.

with MC (i.e., hold-out-set estimation)[3] for the same computational cost. We then choose a larger N and show that the accuracy in estimating the error by using our expressions with Monte Carlo is always greater than by directly using MC for the same computational cost. In fact, the accuracy of using the expressions is never worse than MC even when MC is executed for 10 times the number of iterations as those of our expressions. The true error or the golden standard against which we compare the accuracy of these estimators in this scenario, (since the expressions are also approximated) is MC that is run for around 200 times the number of iterations as those of the expressions. Moreover, in Tables 1 and 2 we depict the upper bounds on the error as computed using Breiman's strength and correlation based upper bound formula (Breiman, 2001).

4.1 Notation

In the experiments, AF refers to the estimates obtained by using the expressions in conjunction with Monte Carlo. MC-i refers to simple Monte Carlo being executed for i times the number of iterations as those of the expressions. Writing just MC denotes MC-1. The term True Error or TE refers to the golden standard against which we compare AF and MC-i. This is relevant only for large N in experiments on synthetic data and experiments on real data, since AF is itself the golden standard for synthetic data experiments with a small N.

4.2 General Setup

We perform empirical studies on synthetic as well as real data. The experimental setup for synthetic data is as follows: In our initial experiments we fix N to a 100 and then increase it to 10000. The number of classes is fixed to two. We observe the behavior of the error for the three kinds of trees with the number of attributes fixed to $d = 5$ and each attribute having 2 attribute values. We then increase the number of attribute values to 3, to observe the effect that increasing the number of split points has on the performance of the estimators. We also increase the number of attributes to $d = 8$ to study the effect that increasing the number of attributes has on the performance. With this we have a $d+1$ dimensional contingency table whose d dimensions are the attributes and the $(d+1)^{th}$ dimension represents the class labels. When each attribute has two values the total number of cells in the table is $c = 2^{d+1}$ and with three values the total number of cells is $c = 3^d \times 2$. If we fix the probability of observing a datapoint in cell i to be p_i such that $\sum_{i=1}^{c} p_i = 1$ and the sample size to N the distribution that perfectly models this scenario is a multinomial distribution

3. In hold-out set we build a tree, find the test error by averaging over multiple test sets. Perform this procedure multiple times to obtain multiple test errors and find the average and variance of these test errors.

with parameters N and the set $\{p_1, p_2, ..., p_c\}$. In fact, irrespective of the value of d and the number of attribute values for each attribute the scenario can be modeled by a multinomial distribution. In the studies that follow the p_i's are varied and the amount of dependence between the attributes and the class labels is computed for each set of p_i's using the Chi-square test (Connor-Linton, 2003). More precisely, we sum over all i the squares of the difference of each p_i with the product of its corresponding marginals, with each squared difference being divided by this product, that is, correlation $= \sum_i \frac{(p_i - p_{im})^2}{p_{im}}$, where p_{im} is the product of the marginals for the i^{th} cell. The behavior of the error for trees with the three aforementioned stopping criteria is seen for different correlation values and for a class prior of 0.5.

In case of real data, we perform experiments on distributions built on three UCI data sets. We split the continuous attributes at the mean of the given data. We thus can form a contingency table representing each of the data sets. The counts in the individual cells divided by the data set size provide us with empirical estimates for the individual cell probabilities (p_i's). Thus, with the knowledge of N (data set size) and the individual p_i's we have a multinomial distribution. Using this distribution we observe the behavior of the error for the three kinds of trees with results being applicable to other data sets that are similar to the original.

In Tables 1 and 2 we see the upper bounds computed using Breiman's formula (Breiman, 2001): $\kappa \frac{(1-s^2)}{s^2}$ where κ is the correlation between the random decision trees in an ensemble and s is the strength of the resultant classifier.[4] Since, we consider only single random decision trees in this paper and not random forests $\kappa = 1$. To compute s we build a tree and calculate the necessary probabilities. Knowing κ and s we find the upper bound on the GE for the particular classifier. Since, we need an estimate of $E_{\mathcal{Z}(N)}[GE(\zeta)]$, we perform the above procedure multiple times thus building multiple trees and computing an upper bound on GE for each. We then average the upper bounds that we have computed and report the result as an estimate of the upper bound on $E_{\mathcal{Z}(N)}[GE(\zeta)]$.

4.3 Observations

In Figure 3 we observe the behavior of MC vs AF (the golden standard) for $N = 100$. We observe that the estimates provided by MC are reasonable but not as accurate as AF for the same computational cost. The behavior of MC becomes worse as we increase the data set size (N) to 10000 as we discuss now. Figure 4 depicts the error of Fixed height trees for different dimensionalities (5 and 8) and for different number of splits (binary and ternary). We observe here that AF is significantly more accurate than both MC-1 and MC-10. In fact the performance of the 3 estimators namely, AF, MC-1 and MC-10 remains more or less unaltered even with changes in the number of attributes and in the number of splits per attribute. A similar trend is seen for both purity based trees Figure 5 as well as scarcity based trees 6. Though in the case of purity based trees the performance of both MC-1 and MC-10 is much superior as compared with their performance on the other two kinds of trees, especially at low correlations. The reason for this being that, at low correlations the probability in each cell of the multinomial is non-negligible and with $N = 10000$ the event that every cell contains at least a single datapoint is highly likely. Hence, the trees we obtain with high probability using the purity based stopping criteria are all ATT's. Since in an ATT all the leaves are identical irrespective of the ordering of the attributes in any path, the randomness in the classifiers produced, is only due to the randomness in the data generation process and not because of the random attribute selection method. Thus, the space of classifiers over which the error is computed reduces and MC performs

4. For further details refer to Breiman (2001) and Buttrey and Kobayashi (2003).

well even for a relatively fewer number of iterations. At higher correlations and for the other two kinds of trees the probability of smaller trees is reasonable and hence MC has to account for a larger space of classifiers induced by not only the randomness in the data but also by the randomness in the attribute selection method.

In case of real data too Figure 7, the performance of the expressions is significantly superior as compared with MC-1 and MC-10. The performance of MC-1 and MC-10 for the purity based trees is not as impressive here since the data set sizes are much smaller (in the tens or hundreds) compared to 10000 and hence the probability of having an empty cell are not particularly low. Moreover, the correlations are reasonably high (above 0.6).

By inspecting Tables 1 and 2 it is immediately apparent that the bound in Breiman (2001) when applied to a single tree is ineffective in most situations—the prediction for the GE is larger than 1. For this formula to provide reasonable predictions, a large number of mostly uncorrelated trees needs to be used so that the constant κ balances the influence of s.

4.4 Reasons for Superior Performance of Expressions

With simple MC, trees have to be built while performing the experiments. Since, the expectations are over all possible classifiers, that is, over all possible data sets and all possible randomizations in the attribute selection phase, the exhaustive space over which direct MC has to run is huge. No tree has to be explicitly built when using the expressions. Moreover, the probabilities for each path can be computed parallelly. Another reason as to why calculating the moments using expressions works better is that the portion of the probabilities for each path that depend on the attribute selection method are computed *exactly* (i.e., with no error) by the given expressions and the inaccuracies in the estimates only occur due to the sample dependent portion in the probabilities.

5. Discussion

In the previous sections we derived the analytical expressions for the moments of the GE of decision trees and depicted interesting behavior of RDT's built under the 3 stopping criteria. It is clear that using the expressions we obtain highly accurate estimates of the moments of errors for situations of interest. In this section we discuss issues related to extension of the analysis to other attribute selection methods and issues related to computational complexity of algorithm.

5.1 Extension

The conditions presented for the 3 stopping criteria namely, fixed height, purity and scarcity are applicable irrespective of the attribute selection method. Commonly used deterministic attribute selection methods include those based on Information Gain (IG), Gini Gain (GG), Gain ratio (GR) etc. Given a sample the above metrics can be computed for each attribute. Hence, the above metrics can be implemented as corresponding functions of the sample. For example, in the case of IG we compute the loss in entropy ($qlogq$ where the q's are computed from the sample) by the addition of an attribute as we build the tree. We then compare the loss in entropy of all attributes not already chosen in the path and choose the attribute for which the loss in entropy is maximum. Following this procedure we build the path and hence the tree. To compute the probability of *path exists*, we add these sample dependent conditions in the corresponding probabilities. These conditions account for a particular set of attributes being chosen, in the 3 stopping criteria. In other words, these conditions

quantify the conditions in the 3 stopping criteria that are attribute selection method dependent. Similar conditions can be derived for the other attribute selection methods (attribute with maximum gini gain for GG, attribute with maximum gain ratio for GR) from which the relevant probabilities and hence the moments can be computed. Thus, while computing the probabilities given in Equations 1 and 2 the conditions for *path exists* for these attribute selection methods depend totally on the sample. This is unlike what we observed for the randomized attribute selection criterion where the conditions for *path exists* depending on this randomized criterion, were sample independent while the other conditions in purity and scarcity were sample dependent. Characterizing these probabilities enables us in computing the moments of GE for these other attribute selection methods.

In the analysis that we presented, we assumed that the split points for continuous attributes were determined apriori to tree construction. If the split point selection algorithm is dynamic, that is, the split points are selected while building the tree, then in the *path exists* conditions of the 3 stopping criteria we would have to append an extra condition namely, the split occurs at "this" particular attribute value. In reality, the value of "this" is determined by the values that the samples attain for the specific attribute in the particular data set, which is finite (since data set is finite). Hence, while analyzing we can choose a set of allowed values for "this" for each continuous attribute. Using these updated set of conditions for the 3 stopping criteria the moments of GE can be computed.

Another interesting extension to the current work, in which we customized expressions for RDT's is to extend the analysis to Random Forests. Random Forests are essentially an ensemble of RDT's and the decision to classify a datapoint is based on a majority vote taken from this ensemble. Hence, in the analysis to compute $P_{Z(N)}[\zeta(x)=y]$ (which is the key ingredient in finding the moments), we would have to compute the probability of the event that more than half of the trees classify the input x into class y. The precise details as to how this might be accomplished efficiently is a part of future research.

5.2 Scalability

The time complexity of implementing the analysis is proportional to the product of the size of the input/output space[5] and the number of paths that are possible in the tree while classifying a particular input. To this end, it should be noted that if a stopping criterion is not carefully chosen and applied, then the number of possible trees and hence the number of allowed paths can become exponential in the dimensionality. In such scenarios, studying small or at best medium size trees is feasible. For studying larger trees the practitioner should combine stopping criteria (e.g., pruning bound and fixed height or scarcity and fixed height), that is, combine the conditions given for each individual stopping criteria or choose a stopping criterion that limits the number of paths (e.g., fixed height). Keeping these simple facts in mind and on appropriate usage, the expressions can assist in delving into the statistical behavior of the errors for decision tree classifiers. Further speedup w/o compromising much on accuracy is a challenge for the future.

5.3 Strengths and Limitations of the Applied Methodology

We now discuss the primary advantage and weakness of the approach taken by Statistical Learning Theory (SLT) and our methodology from the point of view of studying classification algorithms. SLT categorizes classification algorithms (actually the more general learning algorithms) into dif-

5. In case of continuous attributes the size of the input/output space is the size after discretization.

ferent classes called Concept Classes. The concept class of a classification algorithm is determined by its Vapnik-Chervonenkis (VC) dimension which is related to the shattering capability of the algorithm. Distribution free bounds on the generalization error of a classifier built using a particular classification algorithm belonging to a concept class are derived in SLT. The bounds are functions of the VC dimension, the sample size and the training error. The strength of this technique is that by finding the VC dimension of an algorithm we can derive error bounds for the classifiers built using this algorithm without ever referring to the underlying distribution. A consequence of the fact that the characterization is general is that the bounds are usually loose (Boucheron et al., 2005; Williamson, 2001) which in turn results in making statements about any particular classifier and hence classification algorithm weak.

The idea behind the methodology pursued in this paper was to define a class of classifiers induced by a given learning algorithm and i.i.d. data of a given size. As a consequence, this class of classifiers is much smaller than the classes considered in SLT. Hence, the characterization of this class is strongly connected to the behavior of the classifiers and hence the classification algorithm (as seen in this paper for RDT's). The downside of our method is the fact that we loose the strength to make generalized statements to the extent that SLT makes, that is, bounds that are distribution independent. While the process of characterizing classification algorithms employing the deployed methodology might be tedious, we believe that it leads to a more precise study of individual learning algorithms.

6. Conclusion

In this paper we have developed a general characterization for computing the moments of the GE for decision trees. In particular we have specifically characterized RDT's for three stopping criteria namely, fixed height, purity and scarcity. Being able to compute moments of GE, allows us to compute the moments of the various validation measures and observe their relative behavior. Using the general characterization, characterizations for specific attribute selection measures (e.g., IG, GG etc.) other than randomized can be developed as described before. As a technical result, we have extended the theory in Dhurandhar and Dobra (2009) to be applicable to randomized classification algorithms; this is necessary if the theory is to be applied to random decisions trees as we did in this paper. The experiments reported in Section 4 had two purposes: (a) portray the manner in which the expressions can be used as an exploratory tool to gain a better understanding of decision tree classifiers, and (b) show that the methodology in Dhurandhar and Dobra (2009) together with the developments in this paper provide can prove to be a superior analysis tool when compared with other techniques such as Monte Carlo and distribution free bounds.

More work needs to be done to explore the possibilities and test the limits of the kind of analysis that we have performed. However, if learning algorithms are analyzed in the manner that we have shown, it would aid us in studying them more precisely, leading to better understanding and improved decision-making in the practice of model selection.

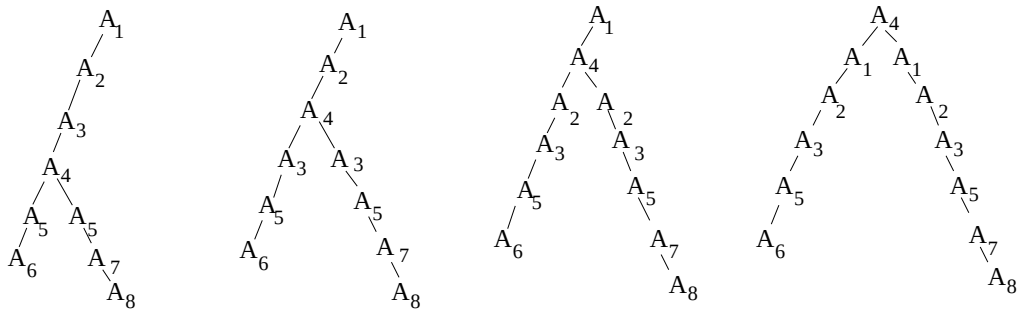


Figure 8: Instances of possible arrangements.

Acknowledgments

We would like to thank Dr. Arunava Banerjee for his comments regarding the introduction. We would also like to thank Dr. Greg Ridgeway and the other anonymous reviewers for their constructive comments. We are grateful to Dr. Medha Dhurandhar for proof reading the paper. This work is supported by the National Science Foundation Grant, NSF-CAREER-IIS-0448264.

Appendix A.

The probability that two paths of lengths l_1 and l_2 ($l_2 \geq l_1$) co-exist in a tree based on the randomized attribute selection method is given by,

$$P[l_1 \textit{ and } l_2 \textit{ length paths co} - \textit{exist}] = \sum_{i=0}^{v} vPr_i (l_1 - i - 1)!(l_2 - i - 1)!(r - v) prob_i$$

where r is the number of attributes common in the two paths, v is the number attributes with the same values in the two paths, $vPr_i = \frac{v!}{(v-i)!}$ denotes permutation and $prob_i = \frac{1}{d(d-1)\ldots(d-i)(d-i-1)^2\ldots(d-l_1+1)^2(d-l_1)\ldots(d-l_2+1)}$.

We now prove the above result. The derivation of the above result will become clearer through the following example. Consider the total number of attributes to be d as usual. Let A_1, A_2 and A_3 be three attributes that are common to both paths and also having the same attribute values. Let A_4 and A_5 be common to both paths but have different attribute values for each of them. Let A_6 belong to only the first path and A_7, A_8 to only the second path. Thus, in our example $l_1 = 6$, $l_2 = 7$, $r = 5$ and $v = 3$. For the two paths to co-exist notice that atleast one of A_4 or A_5 has to be at a lower depth than the non-common attributes A_6, A_7, A_8. This has to be true since, if a non-common attribute say A_6 is higher than A_4 and A_5 in a path of the tree then the other path cannot exist. Hence, in all the possible ways that the two paths can co-exist, one of the attributes A_4 or A_5 has to occur at a maximum depth of $v+1$, that is, 4 in this example. Figure 8a depicts this case. In the successive tree structures, that is, Figure 8b, Figure 8c the common attribute with distinct attribute values (A_4) rises higher up in the tree (to lower depths) until in Figure 8d it becomes the root. To find the probability that the two paths co-exist we sum up the probabilities of such arrangements/tree structures. The probability

of the subtree shown in Figure 8a is $\frac{1}{d(d-1)(d-2)(d-3)(d-4)^2(d-5)^2(d-6)}$ considering that we choose attributes w/o replacement for a particular path. Thus the probability of choosing the root is $\frac{1}{d}$, the next attribute is $\frac{1}{d-1}$ and so on till the subtree splits into two paths at depth 5. After the split at depth 5 the probability of choosing the respective attributes for the two paths is $\frac{1}{(d-4)^2}$, since repetitions are allowed in two separate paths. Finally, the first path ends at depth 6 and only one attribute has to be chosen at depth 7 for the second path which is chosen with a probability of $\frac{1}{d-6}$. We now find the total number of subtrees with such an arrangement where the highest common attribute with different values is at depth of 4. We observe that A_1, A_2 and A_3 can be permuted in whichever way w/o altering the tree structure. The total number of ways of doing this is 3!, that is, $3Pr_3$. The attributes below A_4 can also be permuted in 2!3! w/o changing the tree structure. Moreover, A_4 can be replaced by A_5. Thus, the total number of ways the two paths can co-exist with this arrangement is $3Pr_3 2!3!2$. The probability of the arrangement is hence given by, $\frac{3Pr_3 2!3!2}{d(d-1)(d-2)(d-3)(d-4)^2(d-5)^2(d-6)}$. Similarly, we find the probability of the arrangement in Figure 8b where the common attribute with different values is at depth 3 then at depth 2 and finally at the root. The probabilities for the successive arrangements are $\frac{3Pr_2 3!4!2}{d(d-1)(d-2)(d-3)^2(d-4)^2(d-5)^2(d-6)}$, $\frac{3Pr_1 4!5!2}{d(d-1)(d-2)^2(d-3)^2(d-4)^2(d-5)^2(d-6)}$ and $\frac{3Pr_0 5!6!2}{d(d-1)^2(d-2)^2(d-3)^2(d-4)^2(d-5)^2(d-6)}$ respectively. The total probability for the paths to co-exist is given by the sum of the probabilities of these individual arrangements.

In the general case, where we have v attributes with the same values the number of arrangements possible is $v+1$. This is because the depth at which the two paths separate out lowers from $v+1$ to 1. When the bifurcation occurs at $v+1$ the total number of subtrees is $vPr_v(l_1-v-1)!(l_2-v-1)!(r-v)$ with this arrangement. vPr_v is the permutations of the common attributes with same values. $(l_1-v-1)!$ and $(l_2-v-1)!$ are the total permutations of the attributes in path 1 and 2 respectively after the split. $r-v$ is the number of choices for the split attribute. The probability of any one of the subtrees is $\frac{1}{d(d-1)...(d-v)(d-v-1)^2...(d-l_1+1)^2(d-l_1)...(d-l_2+1)}$ since until a depth of $v+1$ the two paths are the same and then from $v+2$ the two paths separate out. The probability of the first arrangement is thus, $\frac{vPr_v(l_1-v-1)!(l_2-v-1)!(r-v)}{d(d-1)...(d-v)(d-v-1)^2...(d-l_1+1)^2(d-l_1)...(d-l_2+1)}$. For the second arrangement with the bifurcation occurring at a depth of v, the number of subtrees is $vPr_{v-1}(l_1-v)!(l_2-v)!(r-v)$ and the probability of any one of them is $\frac{1}{d(d-1)...(d-v+1)(d-v)^2...(d-l_1+1)^2(d-l_1)...(d-l_2+1)}$. The probability of the arrangement is thus $\frac{vPr_{v-1}(l_1-v)!(l_2-v)!(r-v)}{d(d-1)...(d-v+1)(d-v)^2...(d-l_1+1)^2(d-l_1)...(d-l_2+1)}$. Similarly, the probabilities of the other arrangements can be derived. Hence the total probability for the two paths to co-exist which is the sum of the probabilities of the individual arrangements is given by,

$$P[l_1 \textit{ and } l_2 \textit{ length paths co}-\textit{exist}] = \sum_{i=0}^{v} \frac{vPr_i(l_1-i-1)!(l_2-i-1)!(r-v)}{d(d-1)...(d-i)(d-i-1)^2...(d-l_1+1)^2(d-l_1)...(d-l_2+1)}.$$

References

A. Blum, A. Kalai, and J. Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Computational Learing Theory*, 1999.

S. Boucheron, O. Bousquet, and G. Lugosi. Introduction to statistical learning theory. http://www.kyb.mpg.de/publications/pdfs/pdf2819.pdf, 2005.

L. Breiman. Random forests. http://oz.berkeley.edu/users/breiman/randomforest2001.pdf, 2001.

L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984.

S. Buttrey and I. Kobayashi. On strength and correlation in random forests. In *Proceedings of the 2003 Joint Statistical Meetings, Section on Statistical Computing*, 2003.

J. Connor-Linton. Chi square tutorial. http://www.georgetown.edu/faculty/ballc/webtools/web_chi_tut.html, 2003.

A. Dhurandhar and A. Dobra. Semi-analytical method for analyzing models and model selection measures based on moment analysis. *ACM Transactions on Knowledge Discovery and Data Mining*, 2009.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006. ISSN 0885-6125. doi: http://dx.doi.org/10.1007/s10994-006-6226-1.

M. Hall. Correlation-based feature selection for machine learning. Ph.D diss. Hamilton, NZ: Waikato University, Department of Computer Science, 1998.

M. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE TRANSACTIONS ON KDE*, 2003.

T. Hastie and J. Friedman R. Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *In Proceedings of the Fourteenth IJCAI.*, 1995.

F. Liu, K. Ting, and W. Fan. Maximizing tree diversity by building complete-random decision trees. In *PAKDD*, pages 605–610, 2005.

J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

J. Shao. Linear model selection by cross validation. *JASA*, 88, 1993.

J. Shao. *Mathematical Statistics*. Springer-Verlag, 2003.

L. Smith. A tutorial on principal components analysis. www.csnet.otago.ac.nz/cosc453/ student_tutorials/principal_components.pdf, 2002.

V. Vapnik. *Statistical Learning Theory*. Wiley & Sons, 1998.

R. Williamson. Srm and vc theory (statistical learning theory). http://axiom.anu.edu.au / williams/papers/P151.pdf, 2001.

K. Zhang, W. Fan, B. Buckles, X. Yuan, and Z. Xu. Discovering unrevealed properties of probability estimation trees: On algorithm selection and performance explanation. *ICDM*, 0:741–752, 2006. ISSN 1550-4786. doi: http://doi.ieeecomputersociety.org/10.1109/ICDM.2006.58.

Journal of Machine Learning Research 9 (2008) 2349-2376 Submitted 8/06; Revised 1/08; Published 10/08

Learning to Select Features using their Properties

Eyal Krupka EYAL.KRUPKA@MAIL.HUJI.AC.IL
Amir Navot ANAVOT@GMAIL.COM
Naftali Tishby TISHBY@CS.HUJI.AC.IL
School of Computer Science and Engineering
Interdisciplinary Center for Neural Computation
The Hebrew University Jerusalem, 91904, Israel

Editor: Isabelle Guyon

Abstract

Feature selection is the task of choosing a small subset of features that is sufficient to predict the target labels well. Here, instead of trying to directly determine which features are better, we attempt to learn the properties of good features. For this purpose we assume that each feature is represented by a set of properties, referred to as *meta-features*. This approach enables prediction of the quality of features without measuring their value on the training instances. We use this ability to devise new selection algorithms that can efficiently search for new good features in the presence of a huge number of features, and to dramatically reduce the number of feature measurements needed. We demonstrate our algorithms on a handwritten digit recognition problem and a visual object category recognition problem. In addition, we show how this novel viewpoint enables derivation of better generalization bounds for the joint learning problem of selection and classification, and how it contributes to a better understanding of the problem. Specifically, in the context of object recognition, previous works showed that it is possible to find one set of features which fits most object categories (aka a *universal dictionary*). Here we use our framework to analyze one such universal dictionary and find that the quality of features in this dictionary can be predicted accurately by its meta-features.

Keywords: feature selection, unobserved features, meta-features

1. Introduction

In many supervised learning tasks the input is represented by a very large number of features, many of which are not needed for predicting the labels. *Feature selection* is the task of choosing a small subset of features that is sufficient to predict the target labels well. The main motivations for feature selection are computational complexity, reducing the cost of measuring features, improved classification accuracy and problem understanding. Feature selection is also a crucial component in the context of *feature extraction*. In feature extraction the original input features (for example, pixels) are used to generate new, more complicated features (for example logical AND of sets of 3 binary pixels). Feature extraction is a very useful tool for producing sophisticated classification rules using simple classifiers. One main problem here is that the potential number of additional features one can extract is huge, and the learner needs to decide which of them to include in the model.

In the most common selection paradigm an evaluation function is used to assign scores to subsets of features and a search algorithm is used to search for a subset with a high score. The evaluation

function can be based on the performance of a specific predictor (*wrapper* model) or on some general (typically cheaper to compute) relevance measure of the features to the prediction (*filter* model) (Kohavi and John, 1997). In any case, an exhaustive search over all feature sets is generally intractable due to the exponentially large number of possible sets. Therefore, search methods apply a variety of heuristics, such as hill climbing and genetic algorithms. Other methods simply rank individual features, assigning a score to each feature independently. These methods ignore redundancy and inevitably fail in situations where only a combined set of features is predictive of the target function. However, they are usually very fast, and are very useful in most real-world problems, at least for an initial stage of filtering out useless features. One very common such method is *Infogain* (Quinlan, 1990), which ranks features according to the mutual information[1] between each feature and the labels. Another selection method which we refer to in the following is Recursive Feature Elimination (RFE, Guyon et al., 2002). SVM-RFE is a wrapper selection methods for linear Support Vector Machine (SVM). In each round it measures the quality of the candidate features by training SVM and eliminates the features with the lowest weights. See Guyon and Elisseeff (2003) for a comprehensive overview of feature selection.

In this paper we present a novel approach to the task of feature selection. Classical methods of feature selection tell us which features are better. However, they do not tell us what *characterizes* these features or how to judge new features which were not measured in the training data. We claim that in many cases it is natural to represent each feature by a set of properties, which we call *meta-features*. As a simple example, in image-related tasks where the features are gray-levels of pixels, the (x,y) position of each pixel can be the meta-features. The value of the meta-features is fixed per feature; in other words it is not dependent on the instances. Therefore, we refer to the meta-features as prior knowledge. We use the training set to learn the relation between the meta-feature values and feature usefulness. This in turn enables us to predict the quality of unseen features. This ability is an asset particularly when there are a large number of potential features and it is expensive to measure the value of each feature. For this scenario we suggest a new algorithm called Meta-Feature based Predictive Feature Selection (MF-PFS) which is an alternative to RFE. The MF-PFS algorithm uses predicted quality to select new good features, while eliminating many low-quality features without measuring them. We apply this algorithm to a visual object recognition task and show that it outperforms standard RFE. In the context of object recognition there is an advantage in finding one set of features (referred to as *a universal dictionary*) that is sufficient for recognition of most kinds of objects. Serre et al. (2005) found that such a dictionary can be built by random selection of patches from natural images. Here we show what characterizes good universal features and demonstrate that their quality can be predicted accurately by their meta-features.

The ability to predict feature quality is also a very valuable tool for feature extraction, where the learner has to decide which potential complex features have a good chance of being the most useful. For this task we derive a selection algorithm (called *Mufasa*) that uses meta-features to explore a huge number of candidate features efficiently. We demonstrate the Mufasa algorithm on a handwritten digit recognition problem. We derive generalization bounds for the joint problem of feature selection (or extraction) and classification, when the selection uses meta-features. We show that these bounds are better than those obtained by direct selection.

The paper is organized as follows: we provide a formal definition of the framework and define some notations in Section 2. In Sections 3 and 4 we show how to use this framework to predict the

1. Recall that the mutual information between two random variables X and Y is $I(X;Y) = \sum_{\{x,y\}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$.

quality of individual unseen features and how this ability can be combined with RFE. In Section 5 we apply MF-PFS to an object recognition task. In Section 6 we show how the use of meta-features can be extended to sets of features, and present our algorithm for guiding feature extraction. We illustrate its abilities on the problem of handwritten digit recognition. Our theoretical results are presented in Section 7. We discuss how to choose meta-features wisely in Section 8. Finally we conclude with some further research directions in Section 9. A Matlab code running the algorithms and experiments presented in this paper is available upon request from the authors.

1.1 Related Work

Incorporating prior knowledge about the representation of objects has long been known to profoundly influence the effectiveness of learning. This has been demonstrated by many authors using various heuristics such as specially engineered architectures or distance measures (see, for example, LeCun et al., 1998; Simard et al., 1993). In the context of support vector machines (SVM) a variety of successful methods to incorporate prior knowledge have been published over the last ten years (see, for example, Decoste and Schölkopf 2002 and a recent review by Lauer and Bloch 2006). Krupka and Tishby (2007) proposed a framework that incorporates prior knowledge on features, which is represented by meta-features, into learning. They assume that a weight is assigned to each feature, as in linear discrimination, and they use the meta-features to define a prior on the weights. This prior is based on a Gaussian process and the weights are assumed to be a smooth *function* of the meta-features. While in their work meta-features are used for learning a better classifier, in this work meta-features are used for feature selection.

Taskar et al. (2003) used meta-features of words for text classification when there are features (words) that are unseen in the training set, but appear in the test set. In their work the features are words and the meta-features are words in the neighborhood of each word. They used the meta-features to predict the role of words that are unseen in the training set. Generalization from observed (training) features to unobserved features is discussed by Krupka and Tishby (2008). Their approach involves clustering the instances based on the observed features. What these works and ours have in common is that they all extend learning from the standard instance-label framework to learning in the feature space. Our formulation here, however, is different and allows a mapping of the feature learning problem onto the standard supervised learning framework (see Table 1). Another related model is *Budget Learning* (Lizotte et al., 2003; Greiner, 2005), that explores the issue of deciding which is the most valuable feature to measure next under a limited budget. Other ideas using feature properties to produce or select good features can be found in the literature and have been employed in various applications. For instance, Levi et al. (2004) used this rationale in the context of inductive transfer for object recognition. Raina et al. (2006) also used this approach in the same context for text classification. They use a property of pairs of words which indicates whether they are synonyms or not for the task of estimating the words' covariance matrix. Recently, Lee et al. (2007) used meta-features for feature selection in related tasks. They assume that the meta-features are informative on the relevance of the features. Using a related task they model feature relevance as a function of the meta-features. Kadous and Sammut (2005) used property-based clustering of features for handwritten Chinese recognition and other applications. Our formulation encompasses a more general framework and suggests a systematic way to use the properties as well as derive algorithms and generalization bounds for the combined process of feature selection and classification.

Training set	Features described by meta-features
Test set	Unobserved features
Labels	Feature quality
Hypothesis class	Class of mappings from meta-features to quality
Generalization in feature selection	Predicting the quality of new features
Generalization in the joint problem	Low classification error

Table 1: Feature learning by meta-features as a form of standard supervised learning

2. Framework and Notation

In supervised (classification) learning it is assumed that we have a training set $S^m = \left\{x^i, y^i\right\}_{i=1}^m$, $x^i \in \mathbb{R}^N$ and $y^i = c\left(x^i\right)$ where c is an unknown classification rule. The task is to find a mapping h from $\mathbb{R}^N$ to the label set with a small chance of erring on a new unseen instance, $x \in \mathbb{R}^N$, that was drawn according to the same probability function as the training instances. The N coordinates are called *features*. The standard task of feature selection is to select a subset of features that enables good prediction of the label. This is done by looking for features which are more useful than the others. We can also consider the instances as *abstract entities* in space $\mathcal{S}$ and think of the features as *measurements* on the instances. Thus each feature f can be considered as a function from $\mathcal{S}$ to $\mathbb{R}$; that is, $f : \mathcal{S} \to \mathbb{R}$. We denote the set of all the features by $\{f_j\}_{j=1}^N$. We use the term *feature* to describe both raw input variables (for example, pixels in an image) and variables constructed from the original input variables using some function (for example, product of 3 pixels in the image). We also use F to denote a set of features and S_F^m to denote the training set restricted to F; that is, each instance is described only by the features in F.

Here we assume that each feature is described by a set of k properties $\mathbf{u}(\cdot) = \{u_r(\cdot)\}_{r=1}^k$ which we call *meta-features*. Formally, each $u_r(\cdot)$ is a function from the space of possible measurements to $\mathbb{R}$. Thus each feature f is described by a vector $\mathbf{u}(f) = (u_1(f), \ldots, u_k(f)) \in \mathbb{R}^k$. Note that the meta-features are not dependent on the instances. As already mentioned, and will be described in detail later, this enables a few interesting applications. We also denote a general point in the image of $\mathbf{u}(\cdot)$ by $\mathbf{u}$. log is the base 2 logarithm while ln denotes the natural logarithm. A table that summarizes the above notations and additional notations that will be introduced later appears in Appendix B.

3. Predicting the Quality of Features

In this section we show how meta-features can be used to predict the quality of unseen features. The ability to predict the quality of features without measuring them is advantageous for many applications. In the next section we demonstrate its usage for efficient feature selection for SVM (Vapnik, 1995), when it is very expensive to measure each feature.

We assume that we observe only a subset of the N features; that is in the training set we only see the value of some of the features. We can directly measure the quality (that is, usefulness) of these features using the training set. Based on the quality of these features, our goal is to predict the quality of all features, including features that were not part of the training set. Thus we can think of the training set not only as a "training set of instances", but also as a "training set of features".

More formally, our goal is to use the training set S^m and the set of meta-features to learn a mapping $\hat{Q} : \mathbb{R}^k \longrightarrow \mathbb{R}$ that predicts the quality of a feature using the values of its meta-features.

Algorithm 1 $\hat{Q}$ =quality_map(S^m, featquality, regalg)

1. measure the feature quality vector: $Y_{MF} = \text{featquality}(S^m)$
2. calculate the $N \times k$ meta-features matrix X_{MF}
3. use the regression alg. to learn a mapping from meta-feature value to quality: $\hat{Q} = \text{regalg}(X_{MF}, Y_{MF})$

The quality can be based on any kind of standard evaluation function that uses the labeled training set to evaluate features (for example, Infogain or the square weights assigned by linear SVM). Y_{MF} denotes the vector of measured qualities; that is $Y_{MF}(j)$ is the measured quality of the j's feature in the training set. Now we have a new supervised learning problem, with the **original features** as **instances**, the **meta-features** as **features** and Y_{MF} as the (continuous) target **label**. The analogy to the standard supervised problem is summarized in Table 1. Thus we can use any standard regression learning algorithm to find the required mapping from meta-features to quality. The above procedure is summarized in Algorithm 1. Note that this procedure uses a standard regression learning procedure. That is, the generalization ability to new features can be derived using standard generalization bounds for regression learning. In the next section we give a specific choice for featquality and regalg (see step 2(b) of algorithm 2).

4. Efficient Feature Selection for SVM

Support Vector Machine (SVM) (Vapnik, 1995), is one of the most prominent learning algorithms of the last decade. Many feature selection algorithms for SVM have been suggested (see, for example, Weston et al. 2000). One of the popular methods for linear SVM is Recursive Feature Elimination (Guyon et al., 2002). In SVM-RFE you start by training SVM using all the features, then eliminate the ones with the smallest square weights in the result linear classifier and repeat the same procedure with the remaining features until the set of selected features is small enough. The reason that features are eliminated iteratively and not in one step is that the weights given by SVM to a feature depend on the set of features that was used for training. Thus eliminating only a small number of the worst features in each stage minimizes the unwanted effect of this phenomenon.

The main drawback of SVM-RFE is that all the candidate features have to be measured. This is infeasible when measuring each feature is computationally expensive. We suggest an alternative version of SVM-RFE using meta-features that obviates the need to measure all the features. This algorithm is called Meta-Features based Predictive Feature Selection (MF-PFS). The main idea is to run SVM on only a small (random) subset of the features, and then use the assigned weights for these features to predict the quality of all candidate features using their meta-features (Algorithm 1). Based on this prediction we exclude a group of low quality features, and repeat the process with a smaller set of candidate features. The exact procedure is summarized in Algorithm 2. The suggested algorithm considers all the features while calculating only a small fraction of the them. Thus, it is extremely valuable in a situation where there are a large number of candidate features and the cost of measuring each feature is very high. In the next section we demonstrate MF-PFS on such a data set, and show that it achieves results equivalent to those obtained through standard RFE with an order of magnitude less computation time.

Algorithm 2 $\hat{Q}$ =MF-PFS($S^m, n, t,$ featquality, regalg)

(The algorithm selects n features out of the full set of N)

1. Initialize the set of selected features $F = \left\{ f_j \right\}_{j=1}^{N}$ (all the features).
2. while $|F| > n$,
   (a) Select a set F_0 of random αn features out of F, measure them and produce a training set of features $S^m_{F_0}$.
   (b) Use Algorithm 1 to produce a map $\hat{Q}$ from meta-features values to quality:

$$\hat{Q} = \text{quality\_map}(S^m_{F_0}, \text{featquality}, \text{regalg})$$

   where featquality trains linear SVM and uses the resulting square weights as a measure of quality (Y_{MF} in Algorithm 1). regalg is based on k-Nearest-Neighbor Regression (Navot et al., 2006).
   (c) Use $\hat{Q}$ to estimate the quality of all the features in F.
   (d) Eliminate from F $\min(t|F|, |F| - n)$ features with the lowest estimated quality.

The number of measured features can be further reduced by the following method. The relative number of features we measure in each round (α in step 2(a) of the algorithm) does not have to be fixed; for example, we can start with a small value, since a gross estimation of the quality is enough to eliminate the very worst features, and then increase it in the last stages, where we fine-tune the selected feature set. This way we can save on extra feature measurements without compromising on the performance. Typical values for α might be around 0.5 in the first iteration and slightly above 1 in the last iteration. For the same reason, in step 2(d), it makes sense to decrease the number of features we drop along the iterations. Hence, we adopted the approach that is commonly used in SVM-RFE that drops a constant fraction, $t \in (0,1)$ of the remaining features, where a typical value of t is 0.5. An example of specific choice of parameter values is given in the next section.

5. Experiments with Object Recognition

In this section we use the Caltech-101 data set and adopt the setting in Serre et al. (2005). The data set contains natural images of objects belonging to 101 different categories. The label of each image specifies which object appears in the image, but does not specify the location of the object in the image. Examples of the images are shown in Figure 1. Serre et al. (2007) built a classification system for the above task using linear SVM on a sophisticated representation of the images. In their setting an image is represented by features inspired by the current model of the visual cortex. They show how the features can be built using a hierarchical feedforward system, where each layer mimics the behavior of the relevant layer in the cortex. The interested reader should consult their original paper for all the details on how the features are constructed. Here we simply summarize the description of the features they use, which is sufficient for our needs. First, original images are converted into a representation where the original pixels are replaced by the response to Gabor filters

(Gabor, 1946) of 4 different orientations and 16 different scales, followed by a local maximum of the absolute value over the location, two adjacent scales and decimation (sub-sampling). Thus, each image is replaced by 8 quadruplets of lower resolution images. Each quadruplet corresponds to one scale, and includes all 4 orientations. The following description is over this complex representation. Each feature is defined by a specific patch *prototype.* The prototype is one such quadruplet of a given size (4x4, 8x8, 12x12 or 16x16). The value of the feature on a new image is calculated as follows:

1. The image is converted to the above complex representation.

2. The Euclidean distance of the prototype from every possible patch (that is, at all locations and the 8 scales) of the same size in the image representation is calculated. The minimal distance (over all possible locations and scales) is denoted by d.

3. The value of the feature is $e^{-\beta d^2}$, where β is a parameter.

Step 2 is very computationally costly, and takes by far more resources than any other step in the process. This means that calculating the feature value on all images takes a very long time. Since each feature is defined by a prototype, the feature selection is done by selecting a set of prototypes. In Serre's paper the features are selected randomly by choosing random patches from a data set of natural images (or the training set itself). Namely, to select a prototype you chose an image randomly, convert it to the above representation, then randomly select a patch from it. In Serre's paper, after this random selection, the feature set is fixed, that is no other selection algorithm is used. One method to improve the selected set of features is to use SVM-RFE to select the best features from a larger set of randomly selected candidate features. However, since SVM-RFE requires calculating all the feature values, this option is very expensive to compute. Therefore we suggest choosing the features using meta-features by MF-PFS. The meta-features we use are:

1. The size of the patch (4, 8, 12 or 16).

2. The DC of the patch (average over the patch values).

3. The standard deviation of the patch.

4. The peak value of the patch.

5. Quantiles 0.3 and 0.7 of the values of the patch.

In the following we show that by using these meta-features we can predict the quality of new features with high accuracy, and hence we can drop bad features without measuring their values on the images. This significantly reduces feature selection time. Alternatively, it can improve the classification accuracy since we are able to select from a large set of features in a reasonable training time (using feature selection by MF-PFS). In addition, we draw some interesting observations about the properties of more or less useful features.

Figure 1: Excerpts from the Caltech-101 data-set

5.1 Predicting the Quality of New Features

Since the entire discussion here relies on the assumption that we can predict the quality of a feature from its meta-feature values, we first need to test whether this assumption holds. We need to prove that we can indeed predict the quality of a patch using the above meta-features; that is, from its size, DC, std, peak value and quantile values. We measure feature quality by SVM square weights of multi-class SVM (Shalev-Shwartz and Singer, 2006). Based on this quality definition, Figure 2 presents an example of prototypes of "good" and "bad" features of different sizes. In Figure 3 we explore the quality of features as a function of two meta-features: the patch size and the standard deviation (std). We can see that for small patches (4x4), more details are better, for large patches (16x16) fewer details are better and for medium size patches (for example, 8x8) an intermediate level of complexity is optimal. In other words, the larger the patch, the smaller the optimal complexity. This suggests that by using the size of the patch together with certain measurements of its "complexity" (for example, std) we should be able to predict its quality. In the following experiment we show that such a prediction is indeed possible.

We use Algorithm 1 with the sum square of SVM weights as the direct measure for patch quality and a weighted version of k-Nearest-Neighbor Regression (Navot et al., 2006) as a regressor. To solve the SVM, we use the Shalev-Shwartz and Singer (2006) online algorithm. This algorithm has the advantage of a built-in ability to deal with multi-class problems. We use 500 features as training features. Then we use the map returned by Algorithm 1 to predict the quality of another 500 features. The results presented in Figure 4 show that the prediction is very accurate. The correlation coefficient between measured and predicted quality (on the test set of 500 features) is 0.94. We also assessed the contribution of each meta-feature, by omitting one meta-feature each time and measuring the drop in the correlation coefficient. The meta-features which contributed most to the prediction are size, mean (DC) and std.

The interesting point in the above result is that we show that feature quality can be predicted by its meta-feature values, which represent general statistical properties of the prototype. This observation is notable since it explains the existence of a universal set of features (prototypes) that enables recognition of most objects, regardless of whether the prototypes were taken from pictures that contain the relevant objects or not. Indeed, Serre et al. (2005) found that a set of features (prototypes) which consists of prototypes taken randomly from any natural images constitute such a universal set; however, they did not characterize which features are good. Ullman et al. (2002) also analyzed the properties of good features, where they use a simpler representation of patches (for example, without a Gabor filter). Their conclusion was that intermediate complex features are

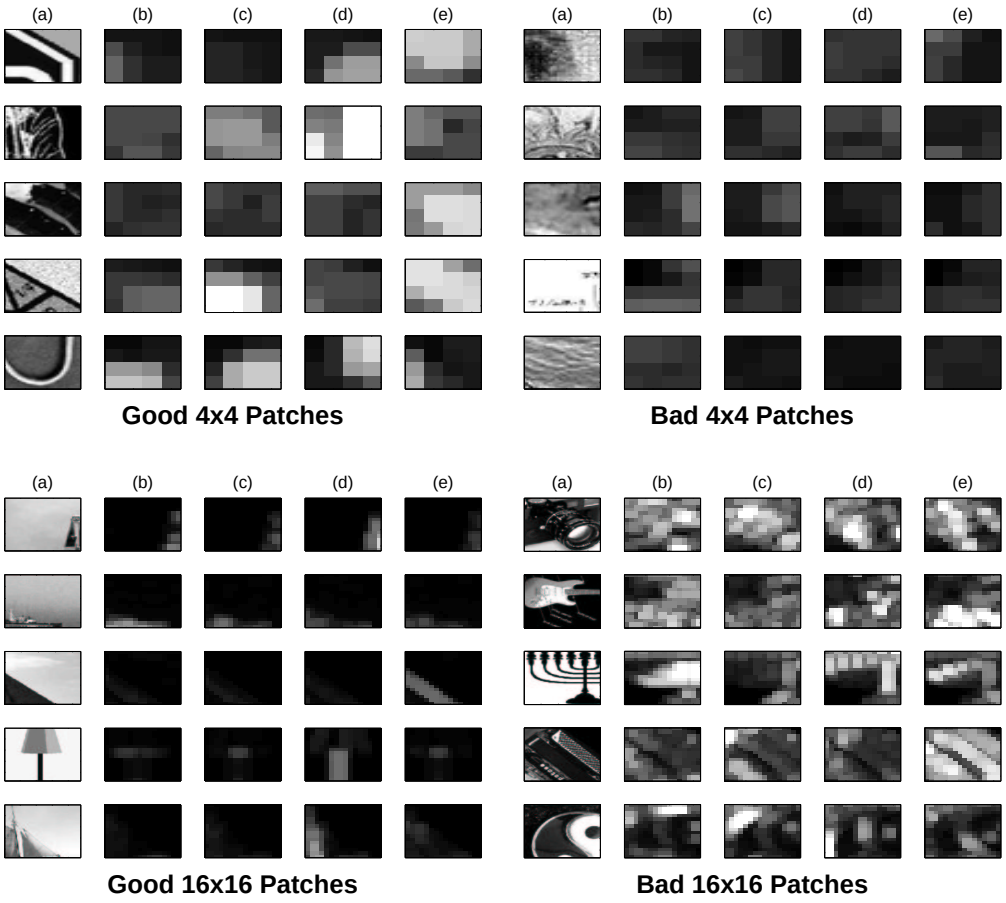


Figure 2: Examples of "good" and "bad" patch prototypes of different sizes. Each row represents one patch. The left column (a) is the image from which the patch was extracted and the other 4 columns (b,c,d,e) correspond to the 4 different orientations (– / | \, respectively). Good small patches are rich in details whereas good large patches are relatively uniform.

the most useful features. However, in their work the features are object fragments, and hence their quality is dependent on the specific training set and not on general statistical properties as we found in this work.

5.2 Applying MF-PFS to Object Recognition

After showing that we are able to predict patches quality, we have a reason to believe that by using MF-PFS (see Section 4) we can obtain an efficient feature selection in Serre et al. (2005) setting. The features in this setting are very expensive to compute and thus a standard selection methods cannot consider many candidate features. MF-PFS on the other hand, allows us to explore a large set of features while measuring only a few of them. In this section we show that MF-PFS (with the above meta-features) indeed succeeds in selecting good features while keeping the computational cost low. Now we turn to present the experimental setup in details. Readers may also skip directly to the results.

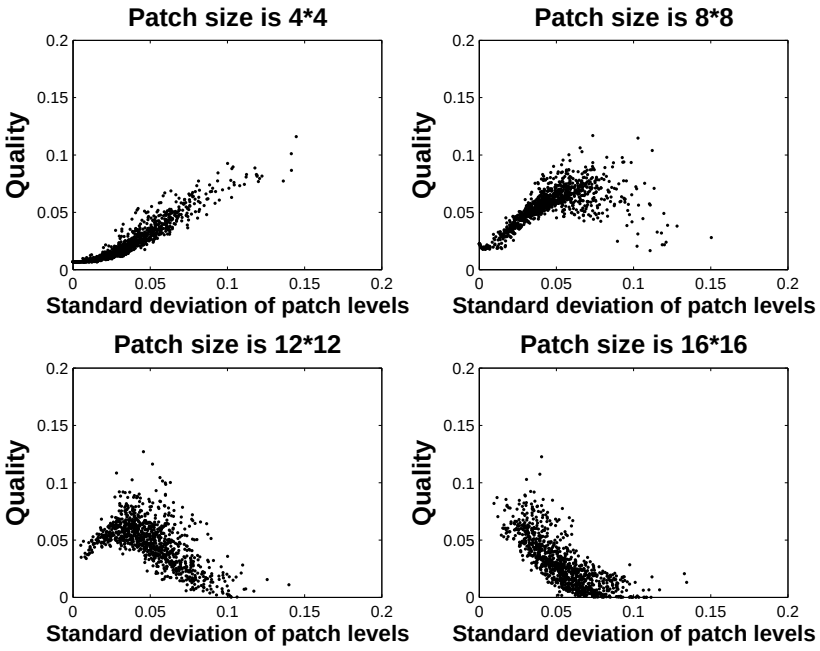


Figure 3: Quality as function of the standard deviation of the meta-feature. Good small patches are "complex" whereas good large patches are "simple".

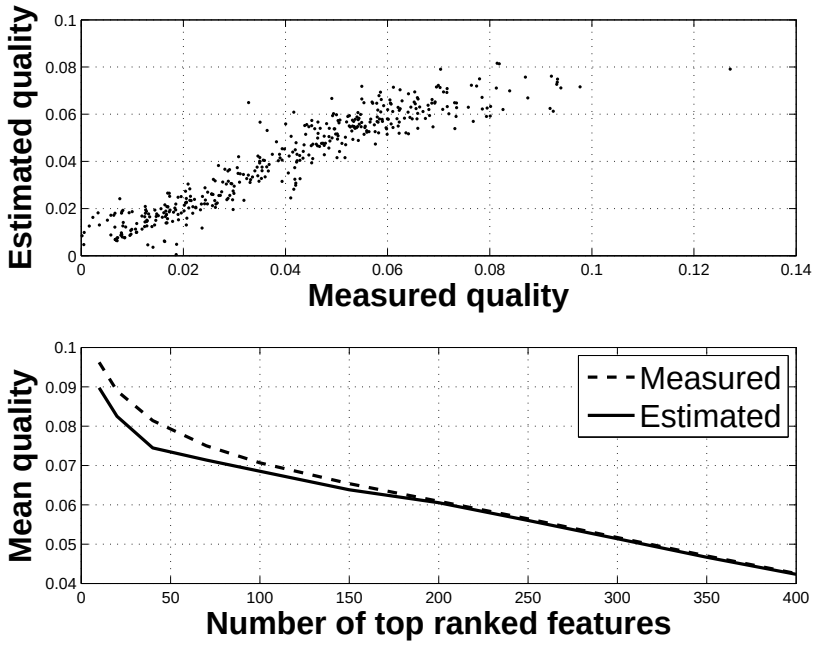


Figure 4: Predicting the quality of patches. top: Scatter plot of the predicted vs. measured quality. The correlation coefficient is 0.94. bottom: The mean quality of the k top ranked features, for different values of k. Ranking using the predicted quality gives (almost) the same mean as ranking by the measured quality.

We use Algorithm 2, with multi-class SVM (Shalev-Shwartz and Singer, 2006) as a classifier and the square of weights as a measure of feature quality. The regression algorithm for quality prediction is based on a weighted version of the *k*-Nearest-Neighbor regression (Navot et al., 2006). Since the number of potential features is virtually infinite, we have to select the initial set of N features.[2] We always start with $N = 10n$ that were selected randomly from natural images in the same manner as in Serre et al. (2005). We use 4 elimination steps (follows from $t = 0.5$, see Algorithm 2). We start with $\alpha = 0.4$ (see step 2(a) of the algorithm) and increase it during the iterations up to $\alpha = 1.2$ in the last step (values in the 2nd and 3rd iteration are 0.8 and 1 respectively). The exact values may seem arbitrary; however the algorithm is not sensitive to small changes in these values. This can be shown as follows. First note that increasing α can only improve the accuracy. Now, note also that the accuracy is bounded by the one achieved by *RFEall*. Thus, since the accuracy we achieve is very close to the accuracy of *RFEall* (see Figure 5), there is a wide range of α selection that hardly affects the accuracy. For example, our initial tests were with $alpha = 0.6, 0.8, 1, 1.2$ which yielded the same accuracy, while measuring almost 10% more features. General guidelines for selecting α are discussed in Section 4. For the above selected value of α, the algorithm measures only about $2.1n$ features out of the $10n$ candidates.

We compared MF-PFS to three different feature selection algorithms. The first was a standard RFE which starts with all the $N = 10n$ features and selects n features using 6 elimination steps (referred to as *RFEall*). The second method was also a standard RFE, but this time it started with $2.1n$ features that were selected randomly from the $N = 10n$ features (referred to as *RFEsmall*). The rationale for this is to compare the performance of our algorithm to standard RFE that measures the same number of features. Since standard RFE does not predict the quality of unseen features, it has to select the initial set randomly. As a baseline we also compared it to random selection of the n features as done in Serre et al. (2005) (referred to as *Baseline*). Note that RFEall considers all the features MF-PFS does, but makes many more measurements (with costs that become infeasible in many cases).[3] On the other hand RFEsmall uses the same number of measurements as MF-PFS, but it considers about one-fifth of the potential features. Finally, in order to estimate the statistical significance of the results we repeated the whole experiment 20 times, with different splits into train instances and test instances.

Results. The results are presented in Figure 5. MF-PFS is nearly as accurate as RFEall, but uses many fewer feature measurements. When RFE measures the same number of features (RFEsmall), it needs to select twice the number of selected features (n) to achieve the same classification accuracy as MF-PFS. Recall that in this setting, as Serre et al. (2005) mentioned, measuring each feature is very expensive; thus these results represent a significant improvement.

6. Guided Feature Extraction

In the previous section we demonstrated the usefulness of meta-features in a scenario where the measurement of each feature is very costly. Here we show how the low dimensional representation of features by a relatively small number of meta-features enables efficient selection even when the number of potential features is very large or even infinite and the evaluation function is expensive (for example, the wrapper model). This is highly relevant to the feature extraction scenario. Note

2. In Section 6 we show that meta-features can be used to explore a space of an infinite number of potential features.
3. It took days on dozens of computers to measure the N=10,000 features on Caltech-101 required for *RFEall*. This is by far the most demanding computational part of the training.

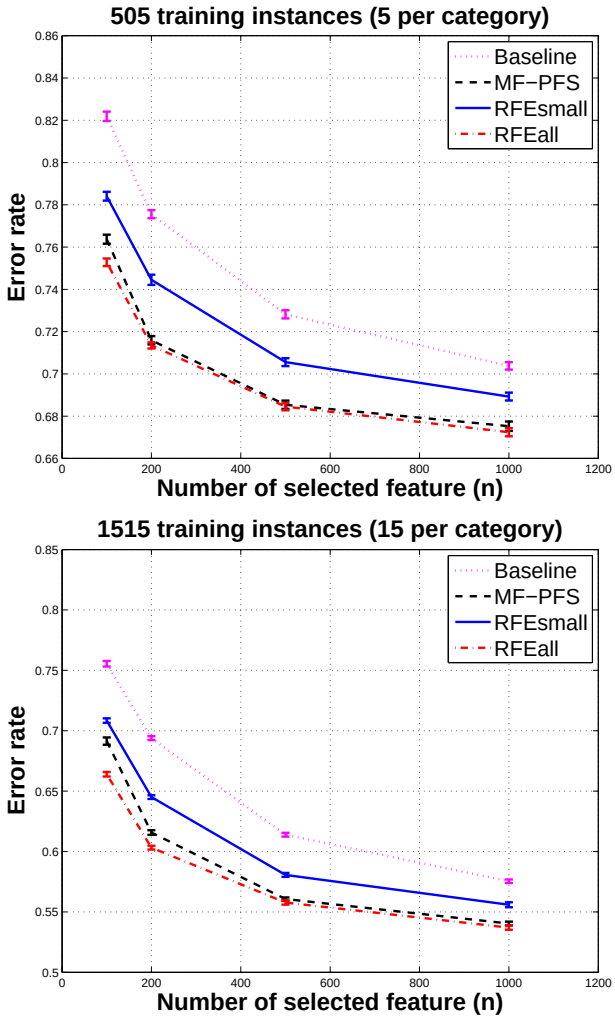


Figure 5: Applying SVM with different feature selection methods for object recognition. When the number of selected features (n) is not too small, our meta-feature based selection (MF-PFS) achieves the same accuracy as RFEall which measures 5 times more features at training time. MF-PFS significantly outperforms RFEsmall which measures the same number of features at training time. To get the same classification accuracy RFEsmall needs about twice the number of features that MF-PFS needs. The results of the baseline algorithm that uses random selection are also presented (Baseline). Error bars show 1-std of the mean performance over the 20 runs.

that an algorithm such as MF-PFS that was presented in the previous sections is not adequate for this scenario, because we need to consider a huge (or even infinite) number of potentially extracted features, and thus even a fast prediction of the quality of all of them is not feasible. Thus we take another approach: a direct search in the meta-feature space, guided by an evaluation of only a subset of representative features.

Algorithm 3 F_{best} =Mufasa(n, J)

1. Initialization: $q_{best} = maxreal,\ \mathbf{u}_0$ is the initial guess of $\mathbf{u}$ (In our experiments we use uniform random).
2. For $j = 1 \ldots J$
   (a) Select (or generate) a new set F_j of n random features according to $p(v|\mathbf{u}_{j-1})$. See Sections 6.1 and 6.2 for details.
   (b) $q_j =$ quality (F_j). (Any measure of quality, In our experiment we use cross-validation classification accuracy of SVM)
   (c) If $q_j \geq q_{best}$
   $F_{best} = F_j,\ \mathbf{u}_{best} = \mathbf{u}_{j-1},\ q_{best} = q_j$
   (d) Randomly select new $\mathbf{u}_j$ which is near $\mathbf{u}_{best}$. For example, in our experiments we add Gaussian random noise to each coordinate of $\mathbf{u}_{best}$, followed by round to the nearest valid value.
3. return F_{best}

6.1 Meta-features Based Search

Assume that we want to select (or extract) a set of n features out of large number of N potential features. We define a stochastic mapping from values of meta-features to selection (or extraction) of features. More formally, let V be a random variable that indicates which feature is selected. We assume that each point $\mathbf{u}$ in the meta-feature space induces density $p(v|\mathbf{u})$ over the features. Our goal is to find a point $\mathbf{u}$ in the meta-feature space such that drawing n features (independently) according to $p(v|\mathbf{u})$ has a high probability of giving us a good set of n features. For this purpose we suggest the *Mufasa* (for Meta-Features Aided Search Algorithm) (Algorithm 3) which implements a stochastic local search in the meta-feature space.

Note that *Mufasa* does not use explicit prediction of the quality of unseen features as we did in MF-PFS, but it is clear that it cannot work unless the meta-features are informative on the quality. Namely, *Mufasa* can only work if the likelihood of drawing a good set of features from $p(v|\mathbf{u})$ is some continuous function of $\mathbf{u}$; that is, a small change in $\mathbf{u}$ results in a small change in the chance of drawing a good set of features. If, in addition, the meta-feature space is "simple"[4] we expect it to find a good point in a small number of steps J. In practice, we can stop when no notable improvement is achieved in a few iterations in a row. The theoretical analysis we present later suggests that overfitting is not a main consideration in the choice of J, since the generalization bound depends on J only logarithmically.

Like any local search over a non-convex target function, convergence to a global optimum is not guaranteed, and the result may depend on the starting point $\mathbf{u}_0$. Note that the random noise added to $\mathbf{u}$ (step 2d in Algorithm 3) may help avoid local maxima. However, other standard techniques to avoid local maxima can also be used (for example, *simulated annealing*). In practice we found, at least in our experiments, that the results are not sensitive to the choice of $\mathbf{u}_0$, though it may affect

4. We elaborate on the meaning of "simple" in Sections 7 and 8.

the number of iterations, J, required to achieve good results. The choice of $p(v|\mathbf{u})$ is application-dependent. In the next section we show how it is done for a specific example of feature generation. For feature selection it is possible to cluster the features based on the similarity of meta-features, and then randomly select features per cluster. Another issue is how to choose the next meta-features point (step 2(d)). Standard techniques for optimizing the step size, such as gradually reducing it, can be used. However, in our experiment we simply added an independent Gaussian noise to each meta-feature.

In Section 6.2 we demonstrate the ability of *Mufasa* to efficiently select good features in the presence of a huge number of candidate (extracted) features on a handwritten digit recognition problem. In Section 7.1 we present a theoretical analysis of *Mufasa*.

6.2 Illustration on a Digit Recognition Task

In this section we use a handwritten digit recognition problem to demonstrate how *Mufasa* works for feature extraction. We used the MNIST (LeCun et al., 1998) data set which contains images of 28×28 pixels of centered digits $(0 \ldots 9)$. We converted the pixels from gray-scale to binary by thresholding. We use extracted features of the following form: logical AND of 1 to 8 pixels (or their negation), which are referred to as *inputs*. This creates intermediate AND-based features which are determined by their input locations. The features we use are calculated by logical OR over a set of such AND-features that are shifted in position in a *shiftInfLen*shiftInvLen* square. For example, assume that an AND-based feature has two inputs in positions (x_1, y_1) and (x_2, y_2), *shiftInvLen=2* and both inputs are taken without negation. Thus the feature value is calculated by OR over 2*2 AND-based features as follows:

$$(Im(x_1,y_1) \wedge Im(x_2,y_2)) \vee (Im(x_1,y_1+1) \wedge Im(x_2,y_2+1)) \vee$$
$$(Im(x_1+1,y_1) \wedge Im(x_2+1,y_2)) \vee (Im(x_1+1,y_1+1) \wedge Im(x_2+1,y_2+1))$$

where $Im(x,y)$ denotes the value of the image in location (x,y). This way we obtain features which are not sensitive to the exact position of curves in the image.

Thus a feature is defined by specifying the set of *inputs*, which inputs are negated, and the value of *shiftInvLen*. Similar features have already been used by Kussul et al. (2001) on the MNIST data set, but with a fixed number of inputs and without shift invariance. The idea of using shift invariance for digit recognition is also not new, and was used, for example, by Simard et al. (1996). It is clear that there are a huge number of such features; thus we have no practical way to measure or use all of them. Therefore we need some guidance for the extraction process, and this is the point where the meta-features framework comes in. We use the following four meta-features:

1. *numInputs*: the number of inputs (1-8).
2. *percentPos*: percent of logic positive pixels (0-100, rounded).
3. *shiftInvLen*: maximum allowed shift value (1-8).
4. *scatter*: average distance of the inputs from their center of gravity (COG) (1-3.5).

In order to find a good value for the meta-features we use *Mufasa* (Algorithm 3) and compare it to some alternatives. A standard method of comparison is to look on the graph of *test error* vs. *the number of selected features* (as done in Figure 5). Here, however, we use a variant of this graph which replaces *the number of selected features* by the total cost of computing the selected features. This modification is required since features with large *shiftInvLen* are significantly more

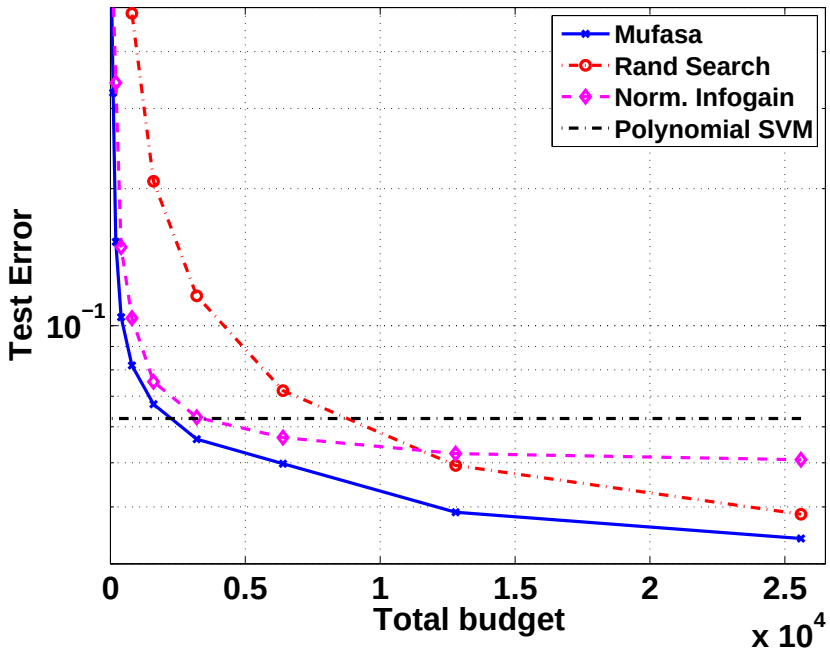


Figure 6: Guided feature extraction for digit recognition. The generalization error rate as a function of the available budget for features, using different selection methods. The number of training instances is 2000 (randomly selected from the MNIST training set). Error bars show a one standard deviation confidence interval. SVM is not limited by the budget, and always implicitly uses all the products of features. We only present the results of SVM with a polynomial kernel of degree 2, the value that gave the best results in this case.

computationally expensive. Thus a selection algorithm is restricted by a *budget* which the total cost of the selected set of features cannot be exceeded, rather than by an allowed number of selected features. We defined the cost of measuring (calculating) a feature as $0.5\left(1+a^2\right)$, where a is the *shiftInvLen* of the feature; this way the cost is proportional to the number of locations where we measure the feature.[5]

We used 2000 images as a training set, and the number of steps, J, is 50. We chose specific features, given a value of meta-features, by re-drawing features randomly from a uniform distribution over the features that satisfied the given value of the meta-features until the full allowed budget was used up. We used 2-fold cross validation of the linear multi-class SVM (Shalev-Shwartz and Singer, 2006; Crammer, 2003) to check the quality of the set of selected features in each step. Finally, for each value of allowed budget we checked the results obtained by the linear SVM on the MNIST standard test set using the selected features.

We compared the results with those obtained using the features selected by Infogain as follows. We first drew features randomly using a budget which was 50 times larger, then we sorted them by Infogain (Quinlan, 1990) normalized by the cost[6] (that is, the value of Infogain divided by

5. The number of inputs does not affect the cost in our implementation since the feature value is calculated by 64-bit logic operations.
6. Infogain without normalization produces worse results.

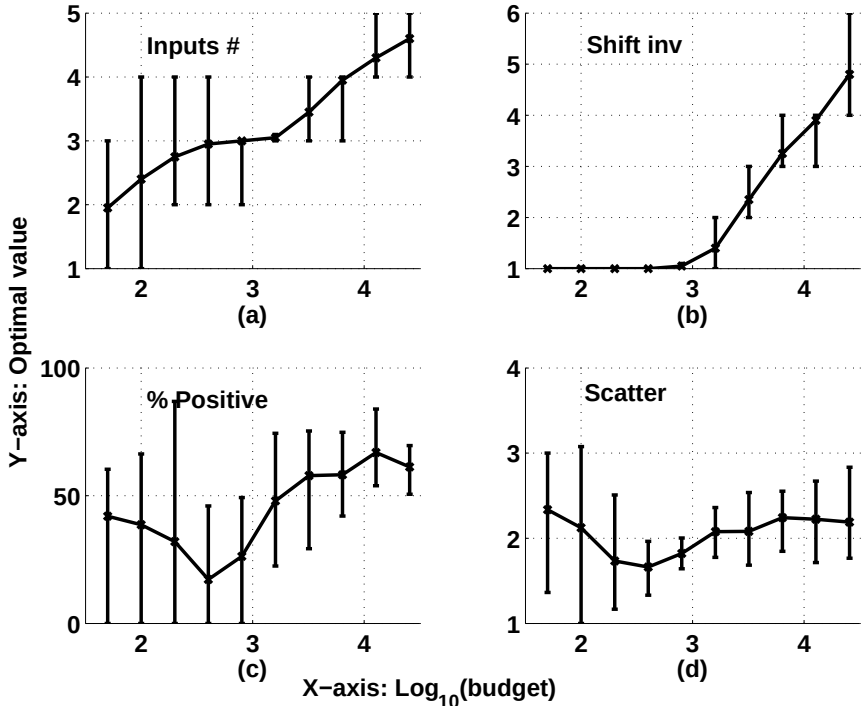


Figure 7: Optimal value of the different meta-features as a function of the budget. The results are averaged over 20 runs, and the error bars indicate the range where values fall in 80% of the runs. The size of the optimal shift invariance and the optimal number inputs increases with the budget.

computational cost of calculating the feature as defined above). We then selected the prefix that used the allowed budget. This method is referred to as *Norm Infogain*. As a sanity check, we also compared the results to those obtained by doing 50 steps of choosing features of the allowed budget randomly; that is, over all possible values of the meta-features. Then we used the set with the lowest 2-fold cross-validation error (referred to as *Rand Search*). We also compared our results to SVM with a polynomial kernel of degree 1-8, that uses the original pixels as input features. This comparison is relevant since SVM with a polynomial kernel of degree k implicitly uses ALL the products of up to k pixels, and the product is equal to AND for binary pixels. To evaluate the statistical significance, we repeated each experiment 20 times, with a different random selection of training sets out of the standard MNIST training set. For the test set, we use the entire $10,000$ test instances of the MNIST data set. The results are presented in Figure 6. It is clear that *Mufasa* outperforms the budget-dependent alternatives, and outperforms SVM for budgets larger than 3000 (about 600 features). It is worth mentioning that our goal here is not to compete with the state-of-art results on MNIST, but to illustrate our concept and to compare the results for the same kind of classifier with and without using our meta-features guided search. Note that our concept can be combined with most kinds of classification, feature selection, and feature extraction algorithms to improve them, as discussed in Section 9.

Another benefit of the meta-features guided search is that it helps understand the problem. To see this we need to take a closer look at the chosen values of the meta-features $(\mathbf{u}_{best})$ as a function of

the available budget. Figure 7 presents the average chosen value of each meta-feature as a function of the budget. As shown in Figure 7b, when the budget is very limited, it is better to take more cheap features rather than fewer more expensive shift invariant features. On the other hand, when we increase the budget, adding these expensive complex features is worth it. We can also see that when the budget grows, the optimal number of inputs increases. This occurs because for a small budget, we prefer features that are less specific, and have relatively high entropy, at the expense of "in class variance". For a large budget, we can permit ourselves to use sparse features (low probability of being 1), but with a gain in specificity. For the scatter meta-features, there is apparently no correlation between the budget and the optimal value. The vertical lines (error bars) represent the range of selected values in the different runs. It gives us a sense of the importance of each meta-feature. A smaller error bar indicates higher sensitivity of the classifier performance to the value of the meta-feature. For example, we can see that performance is sensitive to *shiftInvLen* and relatively indifferent to *percentPos*.

7. Theoretical Analysis

In this section we derive generalization bounds for the combined process of selection and classification when the selection process is based on meta-features. We show that in some cases, these bounds are far better than the bounds that assume each feature can be selected directly. This is because we can significantly narrow the number of possible selections, and still find a good set of features. In Section 7.1 we analyzed the case where the selection is made using *Mufasa* (Algorithm 3). In Section 7.2 we present a more general analysis, which is independent of the selection algorithm, and instead assumes that we have a given class of mappings from meta-features to a selection decision.

7.1 Generalization Bounds for *Mufasa* Algorithm

The bounds presented in this section assume that the selection is made using *Mufasa* (Algorithm 3), but they could be adapted to other meta-feature based selection algorithms. Before presenting the bounds, we need some additional notations. We assume that the classifier that is going to use the selected features is chosen from a hypothesis class $\mathcal{H}_c$ of real valued functions and the classification is made by taking the sign.

We also assume that we have a hypothesis class $\mathcal{H}_{fs}$, where each hypothesis is one possible way to select the n out of N features. Using the training set, our feature selection is limited to selecting one of the hypotheses that is included in $\mathcal{H}_{fs}$. As we show later, if $\mathcal{H}_{fs}$ contains all the possible ways of choosing n out of N features, then we get an unattractive generalization bound for large values of n and N. Thus we use meta-features to further restrict the cardinality (or complexity) of $\mathcal{H}_{fs}$. We have a combined learning scheme of choosing both $h_c \in \mathcal{H}_c$ and $h_{fs} \in \mathcal{H}_{fs}$. We can view this as choosing a single classifier from $\mathcal{H}_{fs} \times \mathcal{H}_c$. In the following paragraphs we analyze the equivalent size of hypothesis space $\mathcal{H}_{fs}$ of *Mufasa* as a function of the number of steps in the algorithm.

For the theoretical analysis, we need to bound the number of feature selection hypotheses *Mufasa* considers. For this purpose, we reformulate *Mufasa* as follows. First, we replace step 2(a) with an equivalent deterministic step. To do so, we add a "pre-processing" stage that generates (randomly) J different sets of features of size n according to $p(v|\mathbf{u})$ for any possible value of the point $\mathbf{u}$ in the meta-feature space.[7] Now, in step 2(a) we simply use the relevant set, according to

7. Later on we generalize to the case of infinite meta-feature space.

the current $\mathbf{u}$ and current j. Namely, in step j we use the jth set of features that was created in the pre-processing stage according to $p(v|\mathbf{u}_j)$, where $\mathbf{u}_j$ is the value of $\mathbf{u}$ in this step. Note that the resulting algorithm is identical to the original *Mufasa*, but this new formulation simplifies the theoretical analysis. The key point here is that the pre-processing is done before we see the training set, and that now *Mufasa* can only select one of the feature sets created in the pre-processing. Therefore, the size of $\mathcal{H}_{fs}$, denoted by $|\mathcal{H}_{fs}|$, is the number of hypotheses created in pre-processing.

The following two lemmas upper bound $|\mathcal{H}_{fs}|$, which is a dominant quantity in the generalization bound. The first one handles the case where the meta-features have discrete values, and there are a relatively small number of possible values for the meta-features. This number is denoted by $|MF|$.

Lemma 1 *Any run of* Mufasa *can be duplicated by first generating $J|MF|$ hypotheses and then running* Mufasa *using these hypotheses alone; that is, using $|\mathcal{H}_{fs}| \leq J|MF|$, where J is the number of iterations made by* Mufasa *and $|MF|$ is the number of different values the meta-features can be assigned.*

Proof

We first generate J random feature sets for each of the $|MF|$ possible values of meta-features. The total number of sets we get is $J|MF|$. We have only J iterations in the algorithm, and we generated J feature sets for each possible value of the meta-features. This guarantees that all the hypotheses required by *Mufasa* are available. ■

Note that in order to use the generalization bound of the algorithm, we cannot only consider the subset of J hypotheses that was tested by the algorithm. This is because this subset of hypotheses is affected by the training set (just as one cannot choose a single hypothesis using the training set, and then claim that the hypothesis space of the classifier includes only one hypothesis). However, from Lemma 1, the algorithm search is within no more than $J|MF|$ feature selection hypotheses that were determined *without* using the training set.

The next lemma handles the case where the cardinality of all possible values of meta-features is large relative to 2^J, or even infinite. In this case we can get a tighter bound that depends on J but not on $|MF|$.

Lemma 2 *Any run of* Mufasa *can be duplicated by first generating 2^{J-1} hypotheses and then running* Mufasa *using only these hypotheses; that is, using $|\mathcal{H}_{fs}| \leq 2^{J-1}$, where J is the number of iterations* Mufasa *performs.*

The proof of this lemma is based on PAC-MDL bounds (Blum and Langford, 2003). Briefly, a codebook that maps between binary messages and hypotheses is built without using the training set. Thus, the generalization bound then depends on the length of the message needed to describe the selected hypothesis. For a fixed message length, the upper bound on the number of hypotheses is 2^l where l is the length of the message in bits.

Proof

Mufasa needs to access a random number generator in steps 2(a) and 2(d). To simplify the proof, we move the random number generation used within *Mufasa* to a pre-processing stage that stores a long vector of random numbers. Thus, every time *Mufasa* needs to access a random number, it will

simply get the next stored random number. After this pre-processing, the feature set, which is the output of *Mufasa*, can be one of 2^{J-1} previously determined sets, since it only depends on the $J-1$ binary decisions in step 2(c) of the algorithm (in the first iteration the decision of step 2(c) is fixed, hence we only have $J-1$ decisions that depend on the training set). Thus, we can generate these 2^{J-1} hypotheses before we see the training set. ■

Using the above PAC-MDL technique, we can also reformulate the last part of the proof by showing that each of the feature-set hypotheses can be uniquely described by $J-1$ binary bits, which describes the decisions in step 2(c). A better generalization bound can be obtained if we assume that in the last steps a new hypothesis will rarely be better than the stored one, and hence the probability of replacing the hypothesis in step 2(c) is small. In this case, we can get a data-dependent bound that usually increases more slowly with the number of iterations (J), since the entropy of the message describing the hypothesis is likely to increase slowly for large J.

To state our theorem we also need the following standard definitions:

Definition 3 *Let $\mathcal{D}$ be a distribution over $\mathcal{S} \times \{\pm 1\}$ and $h : \mathcal{S} \to \{\pm 1\}$ a classification function. We denote by $er_{\mathcal{D}}(h)$ the generalization error of h w.r.t $\mathcal{D}$:*

$$er_{\mathcal{D}}(h) = Pr_{s,y\sim\mathcal{D}}[h(s) \neq y].$$

For a sample $S^m = \{(s_k, y_k)\}_{k=1}^m \in (\mathcal{S} \times \{\pm 1\})^m$ and a constant $\gamma > 0$, the γ-sensitive training error is:

$$\hat{er}_S^{\gamma}(h) = \frac{1}{m} |\{i : h(s_i) \neq y_i\} \quad or \quad (s_i \ \ has \ \ sample\text{-}margin < \gamma), |$$

where the sample-margin measures the distance between the instance and the decision boundary induced by the classifier.

Now we are ready to present the main result of this section:

Theorem 4 *Let $\mathcal{H}_c$ be a class of real valued functions. Let S be a sample of size m generated i.i.d from a distribution $\mathcal{D}$ over $\mathcal{S} \times \{\pm 1\}$. If we choose a set of features using* Mufasa*, with a probability of $1-\delta$ over the choices of S, for every $h_c \in \mathcal{H}_c$ and every $\gamma \in (0,1]$:*

$$er_{\mathcal{D}}(h_c) \leq \hat{er}_S^{\gamma}(h_c) +$$

$$\sqrt{\frac{2}{m}\left(d \ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta}\right) + g(J)\right)},$$

where

- *$d = fat_{\mathcal{H}_c}(\gamma/32)$ and $fat_{\mathcal{H}}(\cdot)$ denotes the fat-shattering dimension of class $\mathcal{H}$ (Bartlett, 1998).*

- $g(J) = \min(J\ln 2, \ln(J|MF|))$ *(where J is the number of steps* Mufasa *makes and $|MF|$ is the number of different values the meta-features can be assigned, if this value is finite, and ∞ otherwise).*

Our main tool in proving the above theorem is the following theorem:

Theorem 5 *(Bartlett, 1998)*

Let $\mathcal{H}$ be a class of real valued functions. Let S be a sample of size m generated i.i.d from a distribution $\mathcal{D}$ over $\mathcal{S} \times \{\pm 1\}$; then with a probability of $1-\delta$ over the choices of S, every $h \in \mathcal{H}$ and every $\gamma \in (0,1]$:

$$er_{\mathcal{D}}(h) \leq \hat{er}_S^{\gamma}(h) + \sqrt{\frac{2}{m}\left(d\ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta}\right)\right)},$$

where $d = fat_{\mathcal{H}}(\gamma/32)$

Proof (of theorem 4)

Let $\left\{F_1, ..., F_{|\mathcal{H}_{fs}|}\right\}$ be all possible subsets of the selected features. From Theorem 5 we know that

$$er_{\mathcal{D}}(h_c, F_i) \leq \hat{er}_S^{\gamma}(h_c, F_i) + \sqrt{\frac{2}{m}\left(d\ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta_{F_i}}\right)\right)},$$

where $er_{\mathcal{D}}(h_c, F_i)$ denotes the generalization error of the selected hypothesis for the fixed set of features F_i.

By choosing $\delta_F = \delta/|\mathcal{H}_{fs}|$ and using the union bound, we get that the probability that there exist F_i ($1 \leq i \leq |\mathcal{H}_{fs}|$) such that the equation below does not hold is less than δ

$$er_{\mathcal{D}}(h_c) \leq \hat{er}_S^{\gamma}(h_c) + \sqrt{\frac{2}{m}\left(d\ln\left(\frac{34em}{d}\right)\log(578m) + \ln\left(\frac{8}{\gamma\delta}\right) + \ln|\mathcal{H}_{fs}|\right)}.$$

Therefore, with a probability of $1-\delta$ the above equation holds for *any* algorithm that selects one of the feature sets out of $\left\{F_1, ..., F_{|H_{fs}|}\right\}$. Substituting the bounds for $|\mathcal{H}_{fs}|$ from Lemma 1 and Lemma 2 completes the proof. ■

An interesting point in this bound is that it is independent of the total number of possible features, N (which may be infinite in the case of feature generation). Nevertheless, it can select a good set of features out of $O\left(2^J\right)$ candidate sets. These sets may be non-overlapping, so the potential number of features that are candidates is $O\left(n2^J\right)$. For comparison, Gilad-Bachrach et al. (2004) gives the same kind of bound but for direct feature selection. Their bound has the same form as our bound, but $g(J)$ is replaced by a term of $O(\ln N)$, which is typically much larger than $J\ln 2$. If we substitute $N = n2^J$, then for the experiment described in Section 6.2 $n\ln N = Jn(\ln 2n) \cong 375000$ while $\ln(J|MF|) \cong 11$.

7.2 VC-dimension of Joint Feature Selection and Classification

In the previous section we presented an analysis which assumes that the selection of features is made using *Mufasa*. In this section we turn to a more general analysis, which is independent of the specific selection algorithm, and rather assumes that we have a given class $\mathcal{H}_s$ of mappings from meta-features to a selection decision. Formally, $\mathcal{H}_s$ is a class of mappings from meta-feature value to $\{0,1\}$; that is, for each $h_s \in \mathcal{H}_s$, $h_s : \mathbb{R}^k \to \{0,1\}$. h_s defines which features are selected as follows:

$$f \text{ is selected} \iff h_s(\mathbf{u}(f)) = 1,$$

where, as usual, $\mathbf{u}(f)$ is the value of the meta-features for feature f. Given the values of the meta-features of all the features together with h_s we get a single feature selection hypothesis. Therefore, $\mathcal{H}_s$ and the set of possible values of meta-features indirectly defines our feature selection hypothesis class, $\mathcal{H}_{fs}$. Since we are interested in selecting exactly n features (n is predefined), we use only a subset of $\mathcal{H}_s$ where we only include functions that imply the selection of n features.[8] For simplicity, in the analysis we use the VC-dim of $\mathcal{H}_s$ without this restriction, which is an upper bound of the VC-dim of the restricted class.

Our goal is to calculate an upper bound on the VC-dimension (Vapnik, 1998) of the joint problem of feature-selection and classification. To achieve this, we first derive an upper bound on $|\mathcal{H}_{fs}|$ as a function of VC-dim$(\mathcal{H}_s)$ and the number of features N.

Lemma 6 *Let $\mathcal{H}_s$ be a class of mappings from the meta-feature space ($\mathbb{R}^k$) to $\{0,1\}$, and let $\mathcal{H}_{fs}$ be the induced class of feature selection schemes; the following inequality holds:*

$$|\mathcal{H}_{fs}| \leq \left(\frac{eN}{\text{VC-dim}(\mathcal{H}_s)}\right)^{\text{VC-dim}(\mathcal{H}_s)}.$$

Proof

The above inequality follows directly from the well known fact that a class with VC-dim d cannot produce more than $\left(\frac{em}{d}\right)^d$ different partitions of a sample of size m (see, for example, Kearns and Vazirani 1994 pp. 57).

■

The next lemma relates the VC dimension of the classification concept class (d_c), the cardinality of the selection class ($|\mathcal{H}_{fs}|$) and the VC-dim of the joint learning problem.

Lemma 7 *Let $\mathcal{H}_{fs}$ be a class of the possible selection schemes for selecting n features out of N and let $\mathcal{H}_c$ be a class of classifiers over $\mathbb{R}^n$. Let $d_c = d_c(n)$ be the VC-dim of $\mathcal{H}_c$. If $d_c \geq 11$ then the VC-dim of the combined problem (that is, choosing $(h_{fs}, h_c) \in \mathcal{H}_{fs} \times \mathcal{H}_c$) is bounded by $(d_c + \log|\mathcal{H}_{fs}| + 1)\log d_c$.*

The proof of this lemma is given in Appendix A.

Now we are ready to state the main theorem of this section.

8. Note that one valid way to define $\mathcal{H}_s$ is by applying a threshold on a class of mappings from meta-feature values to feature quality, $\hat{Q} : \mathbb{R}^k \to \mathbb{R}$. See Example 2 at the end of this section.

Theorem 8 *Let $\mathcal{H}_s$ be a class of mappings from the meta-feature space ($\mathbb{R}^k$) to $\{0,1\}$, let $\mathcal{H}_{fs}$ be the induced class of feature selection schemes for selecting n out of N features and let $\mathcal{H}_c$ be a class of classifiers over $\mathbb{R}^n$. Let $d_c = d_c(n)$ be the VC-dim of the $\mathcal{H}_c$. If $d_c \geq 11$, then the VC-dim of the joint class $\mathcal{H}_{fs} \times \mathcal{H}_c$ is upper bounded as follows*

$$VC\text{-}dim(\mathcal{H}_{fs} \times \mathcal{H}_c) \leq \left(d_c + d_s \log \frac{eN}{d_s} + 1 \right) \log d_c,$$

where d_s is the VC-dim of $\mathcal{H}_s$.

The above theorem follows directly by substituting Lemma 6 in Lemma 7.

To illustrate the gain of the above theorem we calculate the bound for a few specific choices of $\mathcal{H}_s$ and $\mathcal{H}_c$:

1. First, note that if we do not use meta-features, but consider all the possible ways to select n out of N features the above bound is replaced by

$$\left(d_c + \log \binom{N}{n} + 1 \right) \log d_c, \tag{1}$$

   which is very large for reasonable values of N and n.

2. Assuming that both $\mathcal{H}_s$ and $\mathcal{H}_c$ are classes of linear classifiers on $\mathbb{R}^k$ and $\mathbb{R}^n$ respectively, then $d_s = k+1$ and $d_c = n+1$ and we get that the VC of the combined problem of selection and classification is upper bounded by

$$O\left((n + k \log N) \log n \right).$$

   If $\mathcal{H}_c$ is a class of linear classifiers, but we allow any selection of n features the bound is (by substituting in 1):

$$O\left((n + n \log N) \log n \right),$$

   which is much larger if $k \ll n$. Thus in the typical case where the number of meta-features is much smaller than the number of selected features (for example in Section 6.2) the bound for meta-feature based selection is much smaller.

3. Assuming that the meta-features are binary and $\mathcal{H}_s$ is the class of all possible functions from meta-feature to $\{0,1\}$, then $d_s = 2^k$ and the bound is

$$O\left(\left(d_c + 2^k \log N \right) \log d_c \right),$$

   which is still much better than the bound in equation 1 if $k \ll \log n$.

8. Choosing Good Meta-features

At first glance, it might seem that our new setting only complicates the learning problem. One might claim that in addition to the standard hard task of finding a good representation of the instances, now we also have to find a good representation of the features by meta-features. However, our point is that while our setting might be more complicated to understand, in many cases it facilitates

the difficult and crucial problem of finding a good representation. It gives us a systematic way to incorporate prior knowledge about the features in the feature selection and extraction process. It also enables us to use the acquired experience in order to guide the search for good features. The fact that the number of meta-features is typically significantly smaller than the number of features makes it easy to understand the results of the feature selection process. It is easier to guess which properties might be indicative of feature quality than to guess which exact features are good. Nevertheless, the whole concept can work only if we use "good" meta-features, and this requires design. In Sections 5 and 6.2 we demonstrated how to choose meta features for specific problems. In this section we give general guidelines on good choices of meta-features and mappings from meta-feature values to selection of features.

First, note that if there is no correlation between meta-features and quality, the meta-features are useless. In addition, if any two features with the same value of meta-features are redundant (highly correlated), we gain almost nothing from using a large set of them. In general, there is a trade-off between two desired properties:

1. Features with the same value of meta-features have similar quality.

2. There is low redundancy between features with the same value of meta-features.

When the number of features we select is small, we should not be overly concerned about redundancy and rather focus on choosing meta-features that are informative on quality. On the other hand, if we want to select many features, redundancy may be dominant, and this requires our attention. Redundancy can also be tackled by using a distribution over meta-features instead of a single point.

In order to demonstrate the above trade-off we carried out one more experiment using the MNIST data set. We used the same kind of features as in Section 6.2, but this time without shift-invariance and with fixed scatter. The task was to discriminate between 9 and 4 and we used 200 images as the training set and another 200 as the test set. Then we used *Mufasa* to select features, where the meta-features were either the (x,y)-location or the number of inputs. When the meta-feature was the (x,y)-location, the distribution of selecting the features, $p(v|\mathbf{u})$, was uniform in a 4×4 window around the chosen location (step 2a in *Mufasa*). Then we checked the classification error on the test set of a linear SVM (which uses the selected features). We repeated this experiment for different numbers of features.[9] The results are presented in Figure 8. When we use a small number of features, it is better to use the (x,y)-location as a meta-feature whereas when using many features it is better to use the number of inputs as a meta-feature. This supports our contention about the redundancy-homogeneity trade-off. The (x,y)-locations of features are good indicators of their quality, but features from similar positions tend to be redundant. On the other hand, constraints on the number of inputs are less predictive of feature quality but do not cause redundancy.

9. Summary and Discussion

In this paper we presented a novel approach to feature selection. Instead of merely selecting a set of better features out of a given set, we suggest learning the properties of good features. This approach can be used for predicting the quality of features without measuring them even on a single instance.

9. We do not use shift invariance here; thus all the features have the same cost.

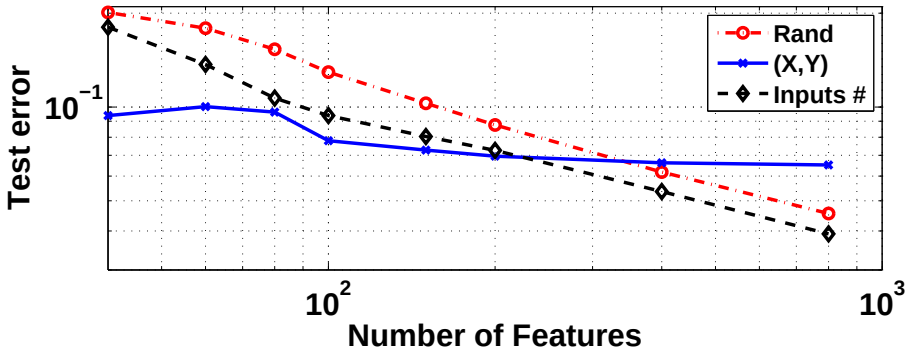


Figure 8: Different choices of meta-features. The generalization error as a function of the number of selected features. The two lines correspond to using different meta-features: (x,y)-location or number of inputs. The results of random selection of features are also presented.

We suggest exploring for new good features by assessing features with meta-feature values similar to those of known good features. Based on this idea, we presented two new algorithms that use feature prediction. The first algorithm is MF-PFS, which estimates the quality of individual features and obviates the need to calculate them on the instances. This is useful when the computational cost of measuring each feature is very high. The second algorithm is *Mufasa*, which efficiently searches for a good feature set without evaluating individual features. *Mufasa* is very helpful in feature extraction, where the number of potential features is huge. Further, it can also help avoiding overfitting in the feature selection task.

In the context of object recognition we showed that the feature (patch) quality can be predicted by its general statistical properties which are not dependent on the objects we are trying to recognize. This result supports the existence of a universal set of features (*universal-dictionary*) that can be used for recognition of most objects. The existence of such a dictionary is a key issue in computer vision and brain research. We also showed that when the selection of features is based on meta-features it is possible to derive better generalization bounds on the combined problem of selection and classification.

In Section 6 we used meta-features to guide feature selection. Our search for good features is computationally efficient and has good generalization properties because we do not examine each individual feature. However, avoiding examination of individual features may also be considered as a disadvantage since we may include some useless individual features. This can be solved by using a meta-features guided search as a fast but rough filter for good features, and then applying more computationally demanding selection methods that examine each feature individually.

In Krupka and Tishby (2007) meta-features were used to build a prior on the weight assigned to each feature by a linear classifier. In that study, the assumption was that meta-features are informative about the feature *weights* (including sign). In this work, however, meta-features should be informative about feature *relevance*, and the exact weight (and sign) is not important. An interesting future research direction would be to combine these concepts of using meta-features into a single framework.

We applied our approach to object recognition and a handwritten digit recognition problem, but we expect our method to be very useful in many other domains. For example, in the problem of

tissue classification according to a gene expression array where each gene is one feature, ontology-based properties may serve as meta-features. In most cases in this domain there are many genes and very few training instances; therefore standard feature selection methods tend to over-fit and thus yield meaningless results (Ein-Dor et al., 2006). A meta-feature based selection can help as it may reduce the complexity of the class of possible selections.

In addition to the applications presented here which involve predicting the quality of unseen features, the meta-features framework can also be used to improve estimation of the quality of features that we do see in the training set. We suggest that instead of using direct quality estimation, we use some regression function on the meta-feature space (as in Algorithm 1). When we have only a few training instances, direct approximation of the feature quality is noisy; thus we expect that smoothing the direct measure by using a regression function of the meta-features may improve the approximation.

Appendix A. A Proof for Lemma 7

Lemma 7 *Let $\mathcal{H}_{fs}$ be a class of the possible selection schemes for selecting n features out of N and let $\mathcal{H}_c$ be a class of classifiers over $\mathbb{R}^n$. Let $d_c = d_c(n)$ be the VC-dim of $\mathcal{H}_c$. If $d_c \geq 11$ then the VC-dim of the combined problem (that is, choosing $(h_{fs}, h_c) \in \mathcal{H}_{fs} \times \mathcal{H}_c$) is bounded by $(d_c + \log|\mathcal{H}_{fs}| + 1)\log d_c$.*

Proof

For a given set of selected features, the possible number of classifications of m instances is upper bounded $\left(\frac{em}{d_c}\right)^{d_c}$ (see Kearns and Vazirani 1994 pp. 57). Thus, for the combined learning problem, the total number of possible classifications of m instances is upper bounded by $|\mathcal{H}_{fs}|\left(\frac{em}{d_c}\right)^{d_c}$. The following chain of inequalities shows that if $m = \left(d_c + \log\left|\mathcal{H}_{fs}\right| + 1\right)\log d_c$ then $\left|\mathcal{H}_{fs}\right|\left(\frac{em}{d_c}\right)^{d_c} < 2^m$:

$$
\begin{aligned}
|\mathcal{H}_{fs}|\left(\frac{e\,(d_c+\log|\mathcal{H}_{fs}|+1)\log d_c}{d_c}\right)^{d_c} &= |\mathcal{H}_{fs}|\,(e\log d_c)^{d_c}\left(1+\frac{\log|\mathcal{H}_{fs}|+1}{d_c}\right)^{d_c} \\
&\leq e\,(|\mathcal{H}_{fs}|)^{1+\log e}\,(e\log d_c)^{d_c} && (2)\\
&\leq (|\mathcal{H}_{fs}|)^{2+\log e}\,(e\log d_c)^{d_c} && (3)\\
&\leq (|\mathcal{H}_{fs}|)^{2+\log e}\,d_c^{d_c+1} && (4)\\
&\leq d_c^{d_c+1}\,(|\mathcal{H}_{fs}|)^{\log d_c} && (5)\\
&= d_c^{d_c+1} d_c^{\left(\log|\mathcal{H}_{fs}|\right)} && (6)\\
&= 2^{\left(d_c+1+\log|\mathcal{H}_{fs}|\right)\log d_c},
\end{aligned}
$$

where we used the following equations / inequalities:

(2) $(1+a/d)^d \leq e^a \qquad \forall a,d>0$

(3) here we assume $|\mathcal{H}_{fs}| > e$, otherwise the lemma is trivial

(4) $(e\log d)^d \leq d^{d+1} \qquad \forall d \geq 1$

(5) $\log d_c > 2$ (since $d_c \geq 11$)

(6) $a^{\log b} = b^{\log a} \qquad \forall a,b > 1$

Therefore, $\left(d_c + \log\left|\mathcal{H}_{fs}\right| + 1\right)\log d_c$ is an upper bound on VC-dim of the combined learning problem.

■

Appendix B. Notation Table

The following table summaries the notation and definitions introduced in the paper for quick reference.

Notation	Short description	Sections
meta-feature	a property that describes a feature	
N	total number of candidate features	
n	number of selected features	
k	number of meta-features	
m	number of training instances	
$\mathcal{S}$	(abstract) instance space	2, 7.1
f	a feature: formally, $f : \mathcal{S} \to \mathbb{R}$	
c	a classification rule	
S^m	S^m is a labeled set of instances (a training set)	2, 3, 7.1
$u_i(f)$	the the value of the$i$'s meta-feature on feature f	2, 6
$\mathbf{u}(f)$	$\mathbf{u}(f) = (u_1(f), \ldots, u_k(f))$, a vector that describes the feature f	2, 6, 7.1
$\mathbf{u}$	a point (vector) in the meta-feature space	2, 6
$\hat{Q}$	a mapping from meta-features value to feature quality	3, 7.2
Y_{MF}	measured quality of the features (for instance Infogain)	3
X_{MF}	$X_{MF}(i,j) =$ the value of the j's meta-feature on the i's feature	3
F, F_j	a set of features	6
V	a random variable indicating which features are selected	6
$p(v\|\mathbf{u})$	the conditional distribution of V given meta-features values ($\mathbf{u}$)	6, 8
h_c	a classification hypothesis	7.1
$\mathcal{H}_c$	the classification hypothesis class	7.1
d_c	the VC-dimension of $\mathcal{H}_c$	7.2
h_{fs}	a feature selection hypothesis - says which n features are selected	7.1
$\mathcal{H}_{fs}$	The feature selection hypothesis class	7.1
h_s	A mapping form meta-feature space to $\{0,1\}$	7.2
$\mathcal{H}_s$	Class of mappings from meta-feature space to $\{0,1\}$	7.2
d_s	The VC-dimension of $\mathcal{H}_s$	
J	number of iteration *Mufasa* (Algorithm 3) does	6, 7.1
$\|MF\|$	Number of possible different values of meta-features	7.1
$er_{\mathcal{D}}(h)$	generalization error of h	7.1
$\hat{er}_S^{\gamma}(h)$	γ-sensitive training error (instance with margin $< \gamma$ count as error)	7.1

References

P.L. Bartlett. The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 1998.

A. Blum and J. Langford. Pac-mdl bounds. *Learning Theory and Kernel Machines*, 2003.

K. Crammer. Mcsvm_1.0: C code for multiclass svm, 2003. http://www.cis.upenn.edu/∼crammer.

D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 2002.

L. Ein-Dor, O. Zuk, and E. Domany. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proceedings of the National Academy of Sciences*, 2006.

D. Gabor. Theory of communication. *J. IEE*, 93:429–459, 1946.

R. Gilad-Bachrach, A. Navot, and N. Tishby. Margin based feature selection - theory and algorithms. In *International Conference on Machine Learning (ICML)*, 2004.

R. Greiner. Using value of information to learn and classify under hard budgets. In *NIPS Workshop on Value of Information in Inference, Learning and Decision-Making*, 2005.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 2003.

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46, 2002.

M. W. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning*, 2005.

M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994.

R. Kohavi and G.H. John. Wrapper for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997.

E. Krupka and N. Tishby. Generalization from observed to unobserved features by clustering. *Journal of Machine Learning Research*, 2008.

E. Krupka and N. Tishby. Incorporating prior knowledge on features into learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

E. Kussul, T. Baidyk, L. Kasatkina, and V. Lukovich. Rosenblatt perceptrons for handwritten digit recognition. In *Int'l Joint Conference on Neural Networks*, pages 1516–20, 2001.

F. Lauer and G. Bloch. Incorporating prior knowledge in support vector machines for classification: a review. *Submitted to Neurocomputing*, 2006.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

S. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *International Conference on Machine Learning (ICML)*, 2007.

K. Levi, M. Fink, and Y. Weiss. Learning from a small number of training examples by exploiting object categories. *LCVPR04 workshop on Learning in Computer Vision*, 2004.

D. Lizotte, O. Madani, and R. Greiner. Budgeted learning of naive-bayes classifiers. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia. Nearest neighbor based feature selection for regression and its application to neural activity. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

J. R. Quinlan. Induction of decision trees. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990.

R. Raina, A.Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proc. Twenty-Third International Conference on Machine Learning*, 2006.

T. Serre, L. Wolf, and T. Poggio. Object recognition with features inspired by visual cortex. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

S. Shalev-Shwartz and Y. Singer. Efficient learning of label ranking by soft projections onto polyhedra. *Journal of Machine Learning Research*, 2006.

P. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, 1996.

P. Y. Simard, Y. A. Le Cun, and Denker. Efficient pattern recognition using a new transformation distance. In *Advances in Neural Information Processing Systems (NIPS)*. 1993.

B. Taskar, M. F. Wong, and D. Koller. Learning on the test data: Leveraging unseen features. In *International Conference on Machine Learning (ICML)*, 2003.

S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 2002.

V. N. Vapnik. *The Nature Of Statistical Learning Theory*. Springer-Verlag, 1995.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

Journal of Machine Learning Research 9 (2008) 2377-2400 Submitted 8/07; Revised 6/08; Published 10/08

Model Selection in Kernel Based Regression using the Influence Function

Michiel Debruyne MICHIEL.DEBRUYNE@UA.AC.BE
Mia Hubert MIA.HUBERT@WIS.KULEUVEN.BE
Department of Mathematics - LStat
K.U.Leuven
Celestijnenlaan 200B, B-3001 Leuven, Belgium

Johan A.K. Suykens JOHAN.SUYKENS@ESAT.KULEUVEN.BE
ESAT-SCD/SISTA
K.U.Leuven
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

Editor: Isabelle Guyon

Abstract

Recent results about the robustness of kernel methods involve the analysis of influence functions. By definition the influence function is closely related to leave-one-out criteria. In statistical learning, the latter is often used to assess the generalization of a method. In statistics, the influence function is used in a similar way to analyze the statistical efficiency of a method. Links between both worlds are explored. The influence function is related to the first term of a Taylor expansion. Higher order influence functions are calculated. A recursive relation between these terms is found characterizing the full Taylor expansion. It is shown how to evaluate influence functions at a specific sample distribution to obtain an approximation of the leave-one-out error. A specific implementation is proposed using a L_1 loss in the selection of the hyperparameters and a Huber loss in the estimation procedure. The parameter in the Huber loss controlling the degree of robustness is optimized as well. The resulting procedure gives good results, even when outliers are present in the data.

Keywords: kernel based regression, robustness, stability, influence function, model selection

1. Introduction

Quantifying the effect of small distributional changes on the resulting estimator is a crucial analysis on many levels. A simple example is leave-one-out which changes the sample distribution slightly by deleting one observation. This leave-one-out error plays a vital role for example in model selection (Wahba, 1990) and in assessing the generalization ability (Poggio et al. 2004 through the concept of stability). Most of these analyses however are restricted to the sample distribution and the addition/deletion of some data points from this sample.

In the field of robust statistics the influence function was introduced in order to analyze the effects of outliers on an estimator. This influence function is defined for continuous distributions that are slightly perturbed by adding a small amount of probability mass at a certain place. In Section 2 some general aspects about the influence function are gathered. Recent results about influence functions in kernel methods include those of Christmann and Steinwart (2004, 2007) for classifica-

tion and regression. In Section 3 these results are stated and their importance is summarized. A new theoretical result concerning higher order influence functions is presented. In Section 4 we show how to evaluate the resulting expressions at sample distributions. Moreover we apply these influence functions in a Taylor expansion approximating the leave-one-out error. In Section 5 we use the approximation with influence functions to select the hyperparameters. A specific implementation is proposed to obtain robustness with a Huber loss function in the estimation step and a L_1 loss in the model selection step. The degree of robustness is controlled by a parameter that can be chosen in a data driven way as well. Everything is illustrated on a toy example and some experiments in Section 6.

2. The Influence Function

In statistics it is often assumed that a sample of data points is observed, all generated independently from the same distribution and some underlying process, but sometimes this is not sufficient. In many applications gathering the observations is quite complex, and many errors or subtle changes can occur when obtaining data. Robust statistics is a branch of statistics that deals with the detection and neutralization of such outlying observations. Roughly speaking a method is called robust if it produces similar results as the majority of observations indicates, no matter how a minority of other observations is placed. A crucial analysis in robust statistics is the behavior of a functional T, not only at the distribution of interest P, but in an entire neighborhood of distributions around P. The influence function measures this behavior. In this section we recall its definition and discuss some links with other concepts.

2.1 Definition

The pioneering work of Hampel et al. (1986) and Huber (1981) considers distributions $P_{\varepsilon,z} = (1-\varepsilon)P + \varepsilon\Delta_z$ where Δ_z denotes the Dirac distribution in the point $z \in \mathcal{X} \times \mathcal{Y}$, representing the contaminated part of the data. For having a robust T, $T(P_{\varepsilon,z})$ should not be too far away from $T(P)$ for any possible z and any small ε. The limiting case of $\varepsilon \downarrow 0$ is comprised in the concept of the influence function.

Definition 1 *Let P be a distribution. Let T be a functional $T : P \to T(P)$. Then the influence function of T at P in the point z is defined as*

$$IF(z;T,P) = \lim_{\varepsilon \to 0} \frac{T(P_{\varepsilon,z}) - T(P)}{\varepsilon}.$$

The influence function measures the effect on the estimator T when adding an infinitesimally small amount of contamination at the point z. Therefore it is a measure of the robustness of T. Of particular importance is the supremum over z. If this is unbounded, then an infinitesimally small amount of contamination can cause arbitrary large changes. For robust estimators, the supremum of its influence function should be bounded. Then small amounts of contamination cannot completely change the estimate and a certain degree of robustness is indeed present. The simplest example is the estimation of the location of a univariate distribution with density f symmetric around 0. The influence function of the mean at $z \in \mathbb{R}$ then equals the function z and is clearly unbounded. If the median of the underlying distribution is uniquely defined, that is if $f(0) > 0$, then the influence function of the median equals $\text{sign}(z)/(2f(0))$ which is bounded. The median is thus more robust than the mean.

2.2 Asymptotic Variance and Stability

From Definition 1 one can see that the influence function is a first order derivative of $T(P_{\varepsilon,z})$ at $\varepsilon = 0$. Higher order influence functions can be defined too:

Definition 2 *Let P be a distribution. Let T be a functional $T : P \to T(P)$. Then the k-th order influence function of T at P in the point z is defined as*

$$IF_k(z;T,P) = \frac{\partial}{\partial^k \varepsilon} T(P_{\varepsilon,z})|_{\varepsilon=0}.$$

If all influence functions exist then the following Taylor expansion holds:

$$T(P_{\varepsilon,z}) = T(P) + \varepsilon IF(z;T,P) + \frac{\varepsilon^2}{2!} IF_2(z;T,P) + \ldots \tag{1}$$

characterizing the estimate at a contaminated distribution in terms of the estimate at the original distribution and the influence functions.

Actually this is a special case of a more general Von Mises expansion (take $Q = P_{\varepsilon,z}$):

$$T(Q) = T(P) + \int IF(x;T,P)d(Q-P)(x) + \ldots$$

Now take Q equal to a sample distribution P_n of a sample $\{z_i\}$ of size n generated i.i.d. from P. Then

$$\begin{aligned} T(P_n) - T(P) &= \int IF(z;T,P)dP_n(z) + \ldots \\ &= \frac{1}{n}\sum_{i=1}^{n} IF(z_i;T,P) + \ldots. \end{aligned}$$

The first term on the right hand side is now a sum of n i.i.d. random variables. If the remaining terms are asymptotically negligible, the central limit theorem thus immediately shows that $\sqrt{n}(T(P_n) - T(P))$ is asymptotically normal with mean 0 and variance

$$ASV(T,P) = \int IF^2(z;T,P)dP(z).$$

Since the asymptotic efficiency of an estimator is proportional to the reciprocal of the asymptotic variance, the integrated squared influence function should be as small as possible to achieve high efficiency. Consider again the estimation of the center of a univariate distribution with density f. At a standard normal distribution the asymptotic variance of the mean equals $\int z^2 dP(z) = 1$, and that of the median equals $\int (\text{sign}(z)/(2f(0)))^2 dP(z) = 1.571$. Thus the mean is more efficient than the median at a normal distribution. However, at a Cauchy distribution for instance, this is completely different: the ASV of the median equals 2.47, but for the mean it is infinite since the second moment of a Cauchy distribution does not exist. Thus to estimate the center of a Cauchy, the median is a much better choice than the mean.

An interesting parallel can be drawn towards the concept of stability in learning theory. Several measures of stability were recently proposed in the literature. The leave-one-out error often plays a vital role, for example in hypothesis stability (Bousquet and Elisseeff, 2001), partial stability (Kutin

and Niyogi, 2002) and CV_{loo}-stability (Poggio et al., 2004). The basic idea is that the result of a learning map T on a full sample should not be very different from the result obtained when removing only one observation. More precisely, let P be a distribution on a set $\mathcal{X} \times \mathcal{Y}$ and $T : P \to T(P)$ with $T(P) : \mathcal{X} \to \mathcal{Y} : x \to T(P)(x)$. Let P_n^{-i} denote the empirical distribution of a sample without the ith observation $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. Poggio et al. (2004) call the map T CV_{loo}-stable for a loss function $L : \mathcal{Y} \to \mathbb{R}^+$ if

$$\lim_{n\to\infty} \sup_{i\in\{1,\ldots,n\}} |L(y_i - T(P_n)(x_i)) - L(y_i - T(P_n^{-i})(x_i))| \to 0 \tag{2}$$

for $n \to \infty$. This means intuitively that the prediction at a point x_i should not be too different whether or not this point is actually used constructing the predictor. If the difference is too large there is no stability, since in that case adding only one point can yield a large change in the result. Under mild conditions it is shown that CV_{loo}-stability is required to achieve good predictions. Let L be the absolute value loss and consider once again the simple case of estimating the location of a univariate distribution. Thus P_n is just a univariate sample of n real numbers $\{y_1, \ldots, y_n\}$. Then the left hand side of (2) equals

$$\lim_{n\to\infty} \sup_{i\in\{1,\ldots,n\}} |T(P_n) - T(P_n^{-i})|.$$

Let $y_{(i)}$ denote the ith order statistic. Consider T the median. Assuming that n is odd and $y_i < y_{(\frac{n+1}{2})}$ (the cases $y_i > y_{(\frac{n+1}{2})}$ and equality can easily be checked as well), we have that

$$|\mathrm{Med}(P_n) - \mathrm{Med}(P_n^{-i})| = \left| y_{(\frac{n+1}{2})} - \frac{1}{2}\left(y_{(\frac{n+1}{2})} + y_{(\frac{n+3}{2})} \right) \right| = \frac{1}{2} |y_{(\frac{n+1}{2})} - y_{(\frac{n+3}{2})}|.$$

If the median of the underlying distribution P is unique, then both $y_{(\frac{n+1}{2})}$ and $y_{(\frac{n+3}{2})}$ converge to this number and CV_{loo} stability is obtained. However, when taking the mean for T, we have that

$$|\mathbb{E}(P_n) - \mathbb{E}(P_n^{-i})| = \left| \frac{1}{n}\sum_{j=1}^{n} y_j - \frac{1}{n-1}\sum_{\substack{j=1 \\ j\neq i}}^{n} y_j \right| = \left| -\frac{1}{n(n-1)}\sum_{\substack{j=1 \\ j\neq i}}^{n} y_j + \frac{y_i}{n} \right|.$$

The first term in this sum equals the sample mean of P_n^{-i} divided by n and thus converges to 0 if the mean of the underlying distribution exists. The second term converges to 0 if

$$\lim_{n\to\infty} \sup_{i\in\{1,\ldots,n\}} \frac{|y_i|}{n} = 0.$$

This means that the largest absolute value of n points sampled from the underlying distribution should not grow too large. For a normal distribution for instance this is satisfied since the largest observation only grows logarithmically: for example the largest of 1000 points generated from a normal distribution only has a very small probability to exceed 5. This is due to the exponentially decreasing density function. For heavy tailed distribution it can be different. A Cauchy density for instance only decreases at the rate of the reciprocal function and $\sup_{i\in\{1,\ldots,n\}} |y_i|$ is of the order $O(n)$. Thus for a normal distribution the mean is CV_{loo} stable, but for a Cauchy distribution it is not.

In summary note that both the concepts of influence function and asymptotic variance on one hand and CV_{loo} stability on the other hand yield the same conclusions: using the sample median as

an estimator is ok as long as the median of the underlying distribution is unique. Then one has CV_{loo} stability and a finite asymptotic variance. Using the sample mean is ok for a normal distribution, but not for a Cauchy distribution (no CV_{loo} stability and an infinite asymptotic variance).

A rigorous treatment of asymptotic variances and regularity conditions can be found in Boos and Serfling (1980) and Fernholz (1983). In any event, it is an interesting link between perturbation analysis through the influence function and variance/efficiency in statistics on one hand, and between leave-one-out and stability/generalization in learning theory on the other hand.

2.3 A Strategy for Fast Approximation of the Leave-one-out Error

In leave-one-out crossvalidation $T(P_n^{-i})$ is computed for every i. This means that the algorithm under consideration has to be executed n times, which can be computationally intensive. If the influence functions of T can be calculated, the following strategy might provide a fast alternative. First note that

$$P_n^{-i} = (1-(\frac{-1}{n-1}))P_n + \frac{-1}{n-1}\Delta_{z_i}.$$

Thus, taking $P_{\varepsilon,z} = P_n^{-i}$, $\varepsilon = -1/(n-1)$ and $P = P_n$, Equation (1) gives

$$T(P_n^{-i}) = T(P_n) + \sum_{j=1}^{\infty}(\frac{-1}{n-1})^j \frac{IF_j(z_i;T,P_n)}{j!}. \tag{3}$$

The right hand side now only depends on the full sample P_n. In practice one can cut off the series after a number of steps ignoring the remainder term, or if possible one can try to estimate the remainder term.

The first goal of this paper is to apply this idea in the context of kernel based regression. Christmann and Steinwart (2007) computed the first order influence function. We will compute higher order terms in (1) and use these results to approximate the leave-one-out estimator applying (3).

3. Kernel Based Regression

In this section we recall some definitions on kernel based regression. We discuss the influence function and provide a theorem on higher order terms.

3.1 Definition

Let $\mathcal{X}, \mathcal{Y}$ be non-empty sets. Denote P a distribution on $\mathcal{X}\times\mathcal{Y} \subseteq \mathbb{R}^d \times \mathbb{R}$. Suppose we have a sample of n observations $(x_i, y_i) \in \mathcal{X}\times\mathcal{Y}$ generated i.i.d. from P. Then P_n denotes the corresponding finite sample distribution. A functional T is a map that maps any distribution P onto $T(P)$. A finite sample approximation is given by $T_n := T(P_n)$.

Definition 3 *A function $K : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is called a kernel on $\mathcal{X}$ if there exists a $\mathbb{R}$-Hilbert space $\mathcal{H}$ and a map $\Phi : \mathcal{X} \to \mathcal{H}$ such that for all $x, x' \in \mathcal{X}$ we have*

$$K(x,x') \;=\; \langle \Phi(x), \Phi(x') \rangle .$$

We call Φ a feature map *and $\mathcal{H}$ a* feature space *of K.*

Frequently used kernels include the linear kernel $K(x_i,x_j) = x_i^t x_j$, polynomial kernel of degree p for which $K(x_i,x_j) = (\tau + x_i^t x_j)^p$ with $\tau > 0$ and RBF kernel $K(x_i,x_j) = \exp(-\|x_i - x_j\|_2^2/\sigma^2)$ with bandwidth $\sigma > 0$. By the reproducing property of $\mathcal{H}$ we can evaluate any $f \in \mathcal{H}$ at the point $x \in \mathcal{X}$ as the inner product of f with the feature map: $f(x) = \langle f, \Phi(x) \rangle$.

Definition 4 *Let K be a kernel function with corresponding feature space $\mathcal{H}$ and let $L : \mathbb{R} \to \mathbb{R}^+$ be a twice differentiable convex loss function. Then the functional $f_{\lambda,K} : P \to f_{\lambda,K}(P) = f_{\lambda,K,P} \in \mathcal{H}$ is defined by*

$$f_{\lambda,K,P} := \underset{f \in \mathcal{H}}{argmin} \mathbb{E}_P L(Y - f(X)) + \lambda \| f \|_{\mathcal{H}}^2$$

where $\lambda > 0$ is a regularization parameter.

The functional $f_{\lambda,K}$ maps a distribution P onto the function $f_{\lambda,K,P}$ that minimizes the regularized risk. When the sample distribution P_n is used, one has that

$$f_{\lambda,K,P_n} := \underset{f \in \mathcal{H}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L(y_i - f(x_i)) + \lambda \| f \|_{\mathcal{H}}^2. \tag{4}$$

Such estimators have been studied in detail, see for example Wahba (1990), Tikhonov and Arsenin (1977) or Evgeniou et al. (2000). In a broader framework (including for example classification, PCA, CCA etc.) primal-dual optimization methodology involving least squares kernel estimators were studied by Suykens et al. (2002b). Possible loss functions include

- the least squares loss: $L(r) = r^2$.
- Vapnik's ε-insensitive loss: $L(r) = \max\{|r| - \varepsilon, 0\}$, with special case the L_1 loss if $\varepsilon = 0$.
- the logistic loss: $L(r) = -\log(4\Lambda(r)[1 - \Lambda(r)])$ with $\Lambda(r) = 1/(1 + e^{-r})$. Note that this is not the same loss function as used in logistic regression.
- Huber loss with parameter $b > 0$: $L(r) = r^2$ if $|r| \leq b$ and $L(r) = 2b|r| - b^2$ if $|r| > b$. Note that the least squares loss corresponds to the limit case $b \to \infty$.

3.2 Influence Function

The following proposition was proven in Christmann and Steinwart (2007).

Proposition 5 *Let $\mathcal{H}$ be a RKHS of a bounded continuous kernel K on $\mathcal{X}$ with feature map $\Phi : \mathcal{X} \to \mathcal{H}$. Furthermore, let P be a distribution on $\mathcal{X} \times \mathcal{Y}$ with finite second moment. Then the influence function of $f_{\lambda,K}$ exists for all $z := (z_x, z_y) \in \mathcal{X} \times \mathcal{Y}$ and we have*

$$IF(z; f_{\lambda,K}, P) = -S^{-1}\left(2\lambda f_{\lambda,K,P}\right) + L'(z_y - f_{\lambda,K,P}(z_x)) S^{-1} \Phi(z_x)$$

where $S : \mathcal{H} \to \mathcal{H}$ is defined by $S(f) = 2\lambda f + \mathbb{E}_P\left[L''(Y - f_{\lambda,K,P}(X)) \langle \Phi(X), f \rangle \Phi(X)\right]$.

Thus if the kernel is bounded and the first derivative of the loss function is bounded, then the influence function is bounded as well. Thus L_1 type loss functions for instance lead to robust estimators. The logistic loss as well since the derivative of this loss function equals $L'(r) = 2 -$

$1/(1+e^{-r})$ which is bounded by 2. For the Huber loss $L'(r)$ is bounded by $2b$. This shows that the parameter b controls the amount of robustness: if b is very large than the influence function can become very large too. For a small b the influence function remains small. For a least squares loss function on the other hand, the influence function is unbounded ($L'(r) = 2r$): the effect of the smallest amount of contamination can be arbitrary large. Therefore it is said that the least squares estimator is not robust.

3.3 Higher Order Influence Functions

For the second order influence function as in Definition 2 the following theorem is proven in the Appendix.

Theorem 6 *Let P be a distribution on $\mathcal{X} \times \mathcal{Y}$ with finite second moment. Let L be a convex loss function that is three times differentiable. Then the second order influence function of $f_{\lambda,K}$ exists for all $z := (z_x, z_y) \in \mathcal{X} \times \mathcal{Y}$ and we have*

$$\begin{aligned} IF_2(z; f_{\lambda,K}, P) = & S^{-1}\Big(2\mathbb{E}_P[IF(z; f_{\lambda,K}, P)(X) L''(Y - f_{\lambda,K}(X))\Phi(X)] \\ & + \mathbb{E}_P[(IF(z; f_{\lambda,K}, P)(X))^2 L'''(Y - f_{\lambda,K,P}(X))] \\ & - 2[IF(z; f_{\lambda,K}, P)(z_x) L''(z_y - f_{\lambda,K}(z_x))\Phi(z_x)] \Big) \end{aligned}$$

where $S : \mathcal{H} \to \mathcal{H}$ is defined by $S(f) = 2\lambda f + \mathbb{E}_P\left[L''(Y - f_{\lambda,K,P}(X))\langle \Phi(X), f \rangle \Phi(X)\right]$.

When the loss function is infinitely differentiable, all higher order terms can in theory be calculated, but the number of terms grows rapidly since all derivatives of L come into play. However, in the special case that all derivatives higher than three are 0, a relatively simple recursive relation exists.

Theorem 7 *Let P be a distribution on $\mathcal{X} \times \mathcal{Y}$ with finite second moment. Let L be a convex loss function such that the third derivative is 0. Then the $(k+1)$th order influence function of $f_{\lambda,K}$ exists for all $z := (z_x, z_y) \in \mathcal{X} \times \mathcal{Y}$ and we have*

$$\begin{aligned} IF_{k+1}(z; f_{\lambda,K}, P) = (k+1) S^{-1}\Big(& \mathbb{E}_P[IF_k(z; f_{\lambda,K}, P)(X) L''(Y - f_{\lambda,K}(X))\Phi(X)] \\ & - [IF_k(z; f_{\lambda,K}, P)(z_x) L''(Z_y - f_{\lambda,K}(z_x))\Phi(z_x)] \Big) \end{aligned}$$

where $S : \mathcal{H} \to \mathcal{H}$ is defined by $S(f) = 2\lambda f + \mathbb{E}_P\left[L''(Y - f_{\lambda,K,P}(X))\langle \Phi(X), f \rangle \Phi(X)\right]$.

4. Finite Sample Expressions

Since the Taylor expansion in (1) is now fully characterized for any distribution P and any z, we can use this to assess the influence of individual points in a sample with sample distribution P_n. Applying Equation (3) with the KBR estimator f_{λ,K,P_n} from (4) we have that

$$f_{\lambda,K,P_n^{-i}}(x_i) = f_{\lambda,K,P_n}(x_i) + \sum_{j=1}^{\infty} \left(\frac{-1}{n-1}\right)^j \frac{IF_j(z_i; f_{\lambda,K}, P_n)(x_i)}{j!}. \tag{5}$$

Let us see how the right hand side can be evaluated in practice.

4.1 Least Squares Loss

First consider taking the least squares loss in (4). Denote Ω the $n \times n$ kernel matrix with i,j-th entry equal to $K(x_i,x_j)$. Let I_n be the $n \times n$ identity matrix and denote $S_n = \Omega/n + \lambda I_n$. The value of f_{λ,K,P_n} at a point $x \in \mathcal{X}$ is given by

$$f_{\lambda,K,P_n}(x) = \frac{1}{n}\sum_{i=1}^{n} \alpha_i K(x_i,x) \quad \text{with} \quad \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix} = S_n^{-1} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{6}$$

which is a classical result going back to Tikhonov and Arsenin (1977). This also means that the vector of predictions in the n sample points simply equals

$$\begin{pmatrix} f_{\lambda,K,P_n}(x_1) \\ \vdots \\ f_{\lambda,K,P_n}(x_n) \end{pmatrix} = H \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{7}$$

with the matrix $H = \frac{1}{n} S_n^{-1}\Omega$, sometimes referred to as the smoother matrix.

To compute the first order influence function at the sample the expression in Proposition 5 should be evaluated at P_n. The operator S at P_n maps by definition any $f \in \mathcal{H}$ onto

$$S_{P_n}(f) = 2\lambda f + \mathbb{E}_{P_n} 2 f(X)\Phi(X) = 2\lambda f + \frac{2}{n}\sum_{j=1}^{n} f(x_j)\Phi(x_j)$$

and thus

$$\begin{aligned} \begin{pmatrix} S_{P_n}(f)(x_1) \\ \vdots \\ S_{P_n}(f)(x_n) \end{pmatrix} &= 2\lambda \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} + \frac{2}{n} \begin{pmatrix} K(x_1,x_1) & \dots & K(x_1,x_n) \\ \vdots & & \\ K(x_n,x_1) & & K(x_n,x_n) \end{pmatrix} \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} \\ &= 2S_n \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix} \end{aligned}$$

which means that the matrix $2S_n$ is the finite sample version of the operator S at the sample P_n. From Proposition 5 it is now clear that

$$\begin{pmatrix} IF(z_i; f_{\lambda,K}, P_n)(x_1) \\ \vdots \\ IF(z_i; f_{\lambda,K}, P_n)(x_n) \end{pmatrix} = S_n^{-1} \left((y_i - f_{\lambda,K,P_n}(x_i)) \begin{pmatrix} K(x_i,x_1) \\ \vdots \\ K(x_i,x_n) \end{pmatrix} - \lambda \begin{pmatrix} f_{\lambda,K,P_n}(x_1) \\ \vdots \\ f_{\lambda,K,P_n}(x_n) \end{pmatrix} \right). \tag{8}$$

In order to evaluate the influence function at sample point z_i at a sample distribution P_n, we only need the full sample fit f_{λ,K,P_n} and the matrix S_n^{-1}, which is already obtained when computing f_{λ,K,P_n} (cf. Equation 6). From Theorem 7 one sees similarly that the higher order terms can be computed

recursively as

$$\begin{pmatrix} IF_{k+1}(z_i; f_{\lambda,K}, P_n)(x_1) \\ \vdots \\ IF_{k+1}(z_i; f_{\lambda,K}, P_n)(x_n) \end{pmatrix} = (k+1) S_n^{-1} \frac{\Omega}{n} \begin{pmatrix} IF(z_i; f_{\lambda,K}, P_n)(x_1) \\ \vdots \\ IF_k(z_i; f_{\lambda,K}, P_n)(x_n) \end{pmatrix} - (k+1) IF_k(z_i; f_{\lambda,K}, P_n)(x_i) S_n^{-1} \begin{pmatrix} K(x_i, x_1) \\ \vdots \\ K(x_i, x_n) \end{pmatrix}. \tag{9}$$

Define $[IFM_k]$ the matrix containing $IF_k(z_j; f_{\lambda,K}, P_n)(x_i)$ at entry i, j. Then (9) is equivalent to

$$[IFM_{k+1}] = (k+1)\left(H\,[IFM_k] - nH \bullet [IFM_k]\right)$$

with $\bullet$ denoting the entrywise matrix product (also known as the Hadamard product). Or equivalently

$$[IFM_{k+1}] = (k+1)\left(H([IFM_k] \bullet M(1-n))\right) \tag{10}$$

with M the matrix containing $1/(1-n)$ at the off-diagonal and 1 at the diagonal. A first idea is now to approximate the series in (5) by cutting it off at some step k:

$$f_{\lambda,K,P_n^{-i}}(x_i) \approx f_{\lambda,K,P_n}(x_i) + \sum_{j=1}^{k} \frac{1}{(1-n)^j j!} [IFM_j]_{i,i}. \tag{11}$$

However using (10) we can do a bit better. Expression (5) becomes

$$\begin{aligned} f_{\lambda,K,P_n^{-i}}(x_i) = f_{\lambda,K,P_n}(x_i) &+ \frac{1}{1-n}[IFM_1]_{i,i} + \frac{1}{1-n}[H(IFM_1 \bullet M)]_{i,i} \\ &+ \frac{1}{1-n}[H(H(IFM_1 \bullet M) \bullet M)]_{i,i} + \ldots \end{aligned}$$

In every term there is a multiplication with H and an entrywise multiplication with M. The latter means that all diagonal elements remain unchanged but the non-diagonal elements are divided by $1-n$. So after a few steps the non-diagonal elements will converge to 0 quite fast. It makes sense to set the non-diagonal elements 0 retaining only the diagonal elements:

$$\begin{aligned} f_{\lambda,K,P_n^{-i}}(x_i) &\approx f_{\lambda,K,P_n}(x_i) + \sum_{j=1}^{k-1} \frac{1}{(1-n)^j j!} [IFM_j]_{i,i} + \frac{1}{(1-n)^k k!} \sum_{j=0}^{\infty} H_{i,i}^j [IFM_k]_{i,i} \\ &= f_{\lambda,K,P_n}(x_i) + \sum_{j=1}^{k-1} \frac{1}{(1-n)^j j!} [IFM_j]_{i,i} + \frac{1}{(1-n)^k k!} \frac{[IFM_k]_{i,i}}{1-H_{i,i}} \end{aligned} \tag{12}$$

since $H_{i,i}$ is always smaller than 1.

4.2 Huber Loss

For the Huber loss function with parameter $b > 0$ we have that

$$L(r) = \begin{cases} r^2 & \text{if } |r| < b. \\ 2b|r| - b^2 & \text{if } |r| > b. \end{cases}$$

and thus

$$L'(r) = \begin{cases} 2r & \text{if } |r| < b \\ 2b \text{ sign}(r) & \text{if } |r| > b \end{cases}, \qquad L''(r) = \begin{cases} 2 & \text{if } |r| < b \\ 0 & \text{if } |r| > b \end{cases}.$$

Note that the derivatives in $|r| = b$ do not exist, but in practice the probability that a residual exactly equals b is 0, so we further ignore this possibility. The following equation holds:

$$f_{\lambda,K,P_n}(x) = \frac{1}{n}\sum_{i=1}^{n} \alpha_i K(x_i, x) \quad \text{with} \quad 2\lambda\alpha_j = L'(y_j - \frac{1}{n}\sum_{i=1}^{n} \alpha_i K(x_i, x_j)). \tag{13}$$

Thus a set of possibly non-linear equations has to be solved in α. Once the solution for the full sample is found, an approximation of the leave-one-out error is obtained in a similar way as for least squares. Proposition 5 for P_n gives the first order influence function.

$$\begin{pmatrix} IF(z_i; f_{\lambda,K}, P_n)(x_1) \\ \vdots \\ IF(z_i; f_{\lambda,K}, P_n)(x_n) \end{pmatrix} = S_b^{-1}\left(L'(y_i - f_{\lambda,K,P_n}(x_i)) \begin{pmatrix} K(x_i, x_1) \\ \vdots \\ K(x_i, x_n) \end{pmatrix} - \lambda \begin{pmatrix} f_{\lambda,K,P_n}(x_1) \\ \vdots \\ f_{\lambda,K,P_n}(x_n) \end{pmatrix} \right)$$

with $S_b = 2\lambda I_n + \Omega \bullet B/n$ and B the matrix containing $L''(y_i - f_{\lambda,K,P_n}(x_i)$ at every entry in the ith column. Let $H_b = S_b^{-1}\Omega/n \bullet B$. Starting from Theorem 7 one finds analogously as (10) the following recursion to compute higher order terms.

$$[IFM_{k+1}] = (k+1)\left(H_b([IFM_k] \bullet M(1-n))\right).$$

Finally one can use these matrices to approximate the leave-one-out estimator as

$$f_{\lambda,K,P_n^{-i}}(x_i) \approx f_{\lambda,K,P_n}(x_i) + \sum_{j=1}^{k-1} \frac{1}{(1-n)^j j!}[IFM_j]_{i,i} + \frac{1}{(1-n)^k k!}\frac{[IFM_k]_{i,i}}{1-[H_b]_{i,i}} \tag{14}$$

in the same way as in (12)

4.3 Reweighted KBR

In Equation (14) the full sample estimator f_{λ,K,P_n} is of course needed. For a general loss function L one has to solve Equation (13) to find f_{λ,K,P_n}. A fast way to do so is to use reweighted KBR with a least squares loss. Let

$$W(r) = \frac{L'(r)}{2r}. \tag{15}$$

Then we can rewrite (13) as

$$2\lambda f_{\lambda,K,P_n}(x_k) = \frac{1}{n}\sum_{i=1}^{n} L'(y_i - f_{\lambda,K,P_n}(x_i))K(x_i,x_k) \quad \forall 1 \le k \le n.$$
$$= \frac{1}{n}\sum_{i=1}^{n} 2W(y_i - f_{\lambda,K,P_n}(x_i))(y_i - f_{\lambda,K,P_n}(x_i))K(x_i,x_k).$$

Denoting $w_i = W(y_i - f_{\lambda,K,P_n}(x_i))$ this means that

$$\lambda f_{\lambda,K,P_n}(x_k) = \frac{1}{n}\sum_{i=1}^{n} w_i y_i K(x_i,x_k) - \frac{1}{n}\sum_{i=1}^{n} w_i f_{\lambda,K,P_n}(x_i)K(x_i,x_k) \quad \forall 1 \le k \le n.$$

Let I_w denote the $n \times n$ diagonal matrix with w_i at entry i,i. Then

$$\begin{pmatrix} f_{\lambda,K,P_n}(x_1) \\ \vdots \\ f_{\lambda,K,P_n}(x_n) \end{pmatrix} = \left(\frac{\Omega}{n} + \lambda I_w\right)^{-1} \frac{\Omega}{n} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{16}$$

and thus f_{λ,K,P_n} can be written as a reweighted least squares estimator with additional weights w_i compared to Equations (6) and (7). Of course these weights still depend on the unknown f_{λ,K,P_n}, so (16) only implicitly defines f_{λ,K,P_n}. It does suggest the following iterative reweighting algorithm.

1. Start with simple least squares computing (7). Denote the solution f^0_{λ,K,P_n}.
2. At step $k+1$ compute weights $w_{i,k} = W(y_i - f^k_{\lambda,K,P_n}(x_i))$.
3. Solve (16) using the weights $w_{i,k}$. Let the solution be f^{k+1}_{λ,K,P_n}.

In Suykens et al. (2002a) it is shown that this algorithm usually converges in very few steps. In Debruyne et al. (2006) the robustness of such stepwise reweighting algorithm is analyzed by calculating stepwise influence functions. It is shown that the influence function is stepwise reduced under certain conditions on the weight function.

For the Huber loss with parameter b Equation (15) means that the corresponding weight function equals $W(r) = 1$ if $|r| \le b$ and $W(r) = b/|r|$ if $|r| > b$. This gives a clear interpretation of this loss function: all observations with error smaller than b remain unchanged, but the ones with error larger than b are downweighted compared to the least squares loss. This also explains the gain in robustness. One can expect better robustness as b decreases.

It would be possible to compute higher order terms of such $k-$step estimators as well. Then one could explicitly use these terms to approximate the leave-one-out error of the $k-$step reweighted estimator. In this paper however we use the reweighting only to compute the full sample estimator f_{λ,K,P_n} and we assume that it is fully converged to the solution of (13). For the model selection (14) is then used.

5. Model Selection

Once the approximation of $f_{\lambda,K,P_n^{-i}}$ is obtained, one can proceed with model selection using the leave-one-out principle. In the next paragraphs we propose a specific implementation taking into account performance as well as robustness.

5.1 Definition

The traditional leave-one-out criterion is given by

$$\mathrm{LOO}(\lambda, K) = \frac{1}{n}\sum_{i=1}^{n} V(y_i - f_{\lambda,K,P_n^{-i}}(x_i)) \tag{17}$$

with V an appropriate loss function. The values of λ and of possible kernel parameters for which this criterion is minimal, are then selected to train the model. The idea we investigate is to replace the explicit leave-one-out by the approximation in (12) for least squares and (14) for the Huber loss.

Definition 8 *The k-th order influence function criterion at a regularization parameter $\lambda > 0$ and kernel K for Huber loss KBR with parameter b is defined as*

$$C_{IF}^k(\lambda, K, b) = \frac{1}{n}\sum_{i=1}^{n} V\left(y_i - f_{\lambda,K,P_n}(x_i) - \sum_{j=1}^{k-1}\frac{1}{(1-n)^j j!}[IFM_j]_{i,i} - \frac{1}{(1-n)^k k!}\frac{[IFM_k]_{i,i}}{1-[H_b]_{i,i}}\right).$$

For KBR with a least squares loss we write

$$C_{IF}^k(\lambda, K, \infty) = \frac{1}{n}\sum_{i=1}^{n} V\left(y_i - f_{\lambda,K,P_n}(x_i) - \sum_{j=1}^{k-1}\frac{1}{(1-n)^j j!}[IFM_j]_{i,i} - \frac{1}{(1-n)^k k!}\frac{[IFM_k]_{i,i}}{1-[H]_{i,i}}\right).$$

since a least squares loss is a limit case of the Huber loss as $b \to \infty$.

Several choices need to be made in practice. For k taking five steps seems to work very well in the experiments. If we refer to the criterion with this specific choice $k = 5$ we write C_{IF}^5. For V one typically chooses the squared loss or the absolute value corresponding to the mean squared error and the mean absolute error. Note that V does not need to be the same as the loss function used to compute f_{λ,K,P_n} (the latter is always denoted by L). Recall that a loss function L with bounded first derivative L' is needed to perform robust fitting. It is important to note that this result following from Proposition 5 holds for a fixed choice of λ and the kernel K. However, if these parameters are selected in a data driven way, outliers in the data might have a large effect on the selection of the parameters. Even if a robust estimator is used, the result could be quite bad if wrong choices are made for the parameters due to the outliers. It is thus important to use a robust loss function V as well. Therefore we set V equal to the absolute value loss function unless we explicitly state differently. In Section 6.1 an illustration is given on what can go wrong if a least squares loss is chosen for V instead of the absolute value.

5.2 Optimizing b

With k and V now specified, the criterion C_{IF}^5 can be used to select optimal hyperparameters for a KBR estimator with L the Huber loss with parameter b. Now the final question remains how to choose b. In Section 4.3 it was argued that b controls the robustness of the estimator since all observations with error smaller than b are downweighted compared to the least squares estimator. Thus we want to choose b small enough such that outlying observations receive sufficiently small weight, but also large enough such that the good non outlying observations are not downweighted too much. A priori it is quite difficult to find such a good choice for b, since this will depend on the scale of the errors.

However, one can also treat b as an extra parameter that is part of the optimization, consequently minimizing C^5_{IF} for λ, K and b simultaneously. The practical implementation we propose is as follows:

1. Let Λ be a set of reasonable values for the regularization parameter λ and let $\mathcal{K}$ be a set of possible choices for the kernel K (for instance a grid of reasonable bandwidths if one considers the RBF kernel).

2. Start with L the least squares loss. Find good choices for λ and K by minimizing $C^5_{IF}(\lambda, K, \infty)$ for all $\lambda \in \Lambda$ and $K \in \mathcal{K}$. Compute the residuals r_i with respect to the least squares fit with these optimal λ and K.

3. Compute a robust estimate of the scale of these residuals. We take the Median Absolute Deviation (MAD):

$$\hat{\sigma}_{err} = \text{MAD}(r_1, \ldots, r_n) = \frac{1}{\Phi^{-1}(0.75)} \text{median}(|r_i - \text{median}(r_i)|) \tag{18}$$

   with $\Phi^{-1}(0.75)$ the 0.75 quantile of a standard normal distribution.

4. Once the scale of the errors is estimated in the previous way, reasonable choices of b can be constructed, for example $\{1,2,3\} \times \hat{\sigma}_{err}$. This means that we compare downweighting observations further away than 1, 2, 3 standard deviations. We also want to compare to the least squares fit and thus set

$$\mathcal{B} = \{\hat{\sigma}_{err}, 2\hat{\sigma}_{err}, 3\hat{\sigma}_{err}, \infty\}.$$

5. Minimize $C^5_{IF}(\lambda, K, b)$ over all $\lambda \in \Lambda$, $K \in \mathcal{K}$ and $b \in \mathcal{B}$. The optimal values of b, λ and K can then be used to construct the final fit.

5.3 Generalized Cross Validation

The criterion C^5_{IF} uses influence functions to approximate the leave-one-out error. Other approximations have been proposed in the literature. In this section we very briefly mention some results that are described for example by Wahba (1990) in the context of spline regression. The following result can be proven.

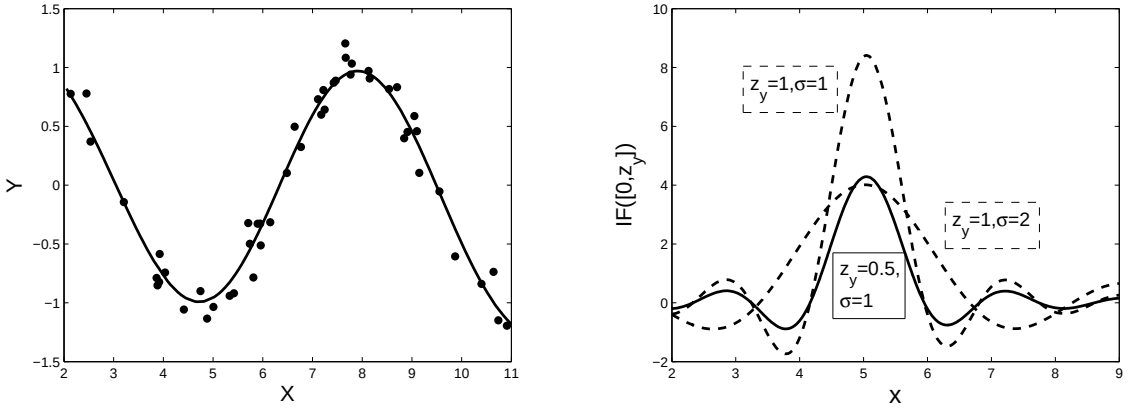
Let $\tilde{P}_n^{-i}$ be the sample P_n with observation (x_i, y_i) replaced by $(x_i, f_{\lambda,K,P_n^{-i}}(x_i))$. Suppose the following conditions are satisfied for any sample P_n:

$$(i) \quad f_{\lambda,K,\tilde{P}_n^{-i}}(x_i) = f_{\lambda,K,P_n^{-i}}(x_i). \tag{19}$$

$$(ii) \quad \text{There exists a matrix } H \text{ such that } \begin{pmatrix} f_{\lambda,K,P_n}(x_1). \\ \vdots \\ f_{\lambda,K,P_n}(x_n) \end{pmatrix} = H \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}. \tag{20}$$

Then

$$f_{\lambda,K,P_n^{-i}}(x_i) = \frac{f_{\lambda,K,P_n}(x_i) - H_{i,i} y_i}{1 - H_{i,i}}. \tag{21}$$



(a) (b)

Figure 1: (a) Data and least squares fit. (b) Influence functions at $[5,0.5]$ with $\sigma = 1$, at $[5,1]$ with $\sigma = 1$ and $\sigma = 2$.

For KBR with the least squares loss condition (22) is indeed satisfied (cf. Equation 7), but condition (19) is not, although it holds approximately. Then (21) can still be used as an approximation of the leave-one-out estimator. The corresponding model selection criterion is given by

$$CV(\lambda,K) = \frac{1}{n}\sum_{i=1}^{n} V\left(\frac{y_i - f_{\lambda,K,P_n}(x_i)}{1-H_{i,i}}\right). \tag{22}$$

We call this approximation CV. Sometimes a further approximation is made replacing every $H_{i,i}$ by $\text{trace}(H)/n$. This is called Generalized Cross Validation (GCV, Wahba, 1990). Note that the diagonal elements of the hatmatrix H play an important role in the approximation with the influence function too (12). Both penalize small values on the diagonal of H.

For KBR with a general loss function one does not have a linear equation of the form of (22), and thus it is more difficult to apply this approximation. We shall thus use CV for comparison in the experiments only in the case of least squares.

6. Empirical Results

We illustrate the results on a toy example and a small simulation study.

6.1 Toy Example

As a toy example 50 data points were generated with x_i uniformly distributed on the interval $[2,11]$ and $y_i = \sin(x_i) + e_i$ with e_i Gaussian distributed noise with standard deviation 0.2. We start with kernel based regression with a least squares loss and a Gaussian kernel. The data are shown in Figure $1(a)$ as well as the resulting fit with $\lambda = 0.001$ and $\sigma = 2$. The first order influence function at $[5,0.5]$ is depicted in Figure $1(b)$ as the solid line. This reflects the asymptotic change in the fit when a point would be added to the data in Figure $1(a)$ at the position $(5,0.5)$. Obviously this influence is the largest at the x-position where we put the outlier, that is, $x = 5$. Furthermore we see that the influence is local, since it decreases as we look further away from $x = 5$. At $x = 8$ for

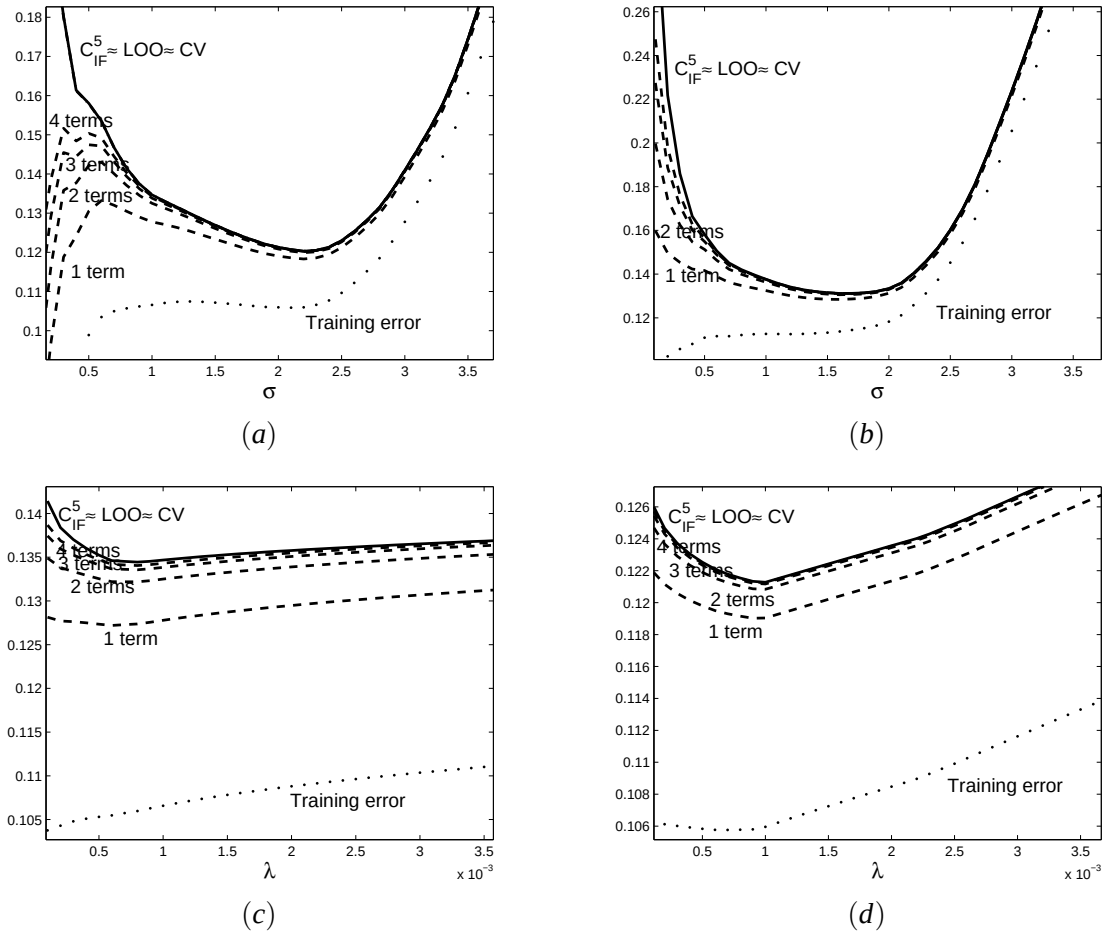


Figure 2: Comparison of training error (dotted line), approximations using (11) (dashed lines), the proposed criterion C_{IF}^k with $k = 5$ (solid line), the exact leave-one-out error and the CV approximation (both collapsing with C_{IF}^k on these plots). Situation (a): as a function of σ at $\lambda = 0.001$, (b) as a function of σ at $\lambda = 0.005$, (c) as a function of λ at $\sigma = 1$, (d) as a function of λ at $\sigma = 2$.

instance the influence function is almost 0. When we change z from $[5, 0.5]$ to $[5, 1]$, the influence function changes too. It still has the same oscillating behavior, but the peaks are now higher. This reflects the non-robustness of the least squares estimator: if we would continue raising the point z, then $IF(z; f_{\lambda,K})$ would become larger and larger, since it is an unbounded function of z. When it comes down to model selection, it is interesting to check the effect of the hyperparameters in play. When we change the bandwidth σ from 1 to 2, the peaks in the resulting influence function in Figure 1 are less sharp and less high. This reflects the loss in stability when small bandwidths are chosen: then the fit is more sensitive to small changes in the data and thus less stable.

Consider now the approximation of the leave-one-out error using the influence functions. We still use the same data as in the previous paragraph. The dashed lines in Figure $2(a)$ show the approximations using (11), that is simply cutting off the expansion after a number of steps, at fixed

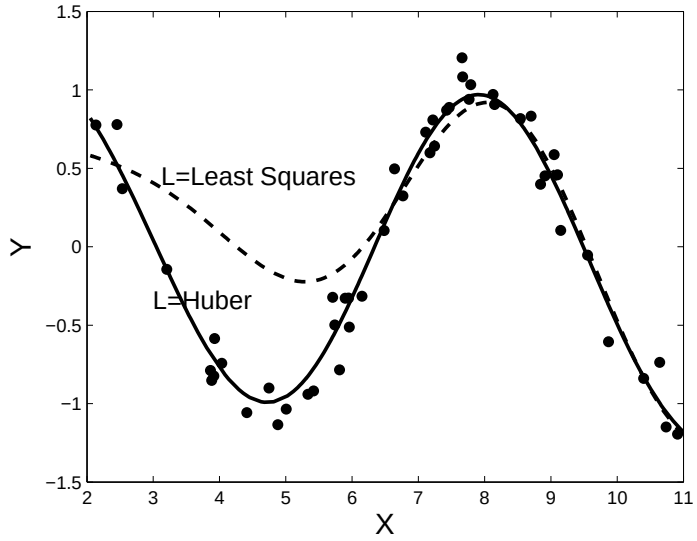
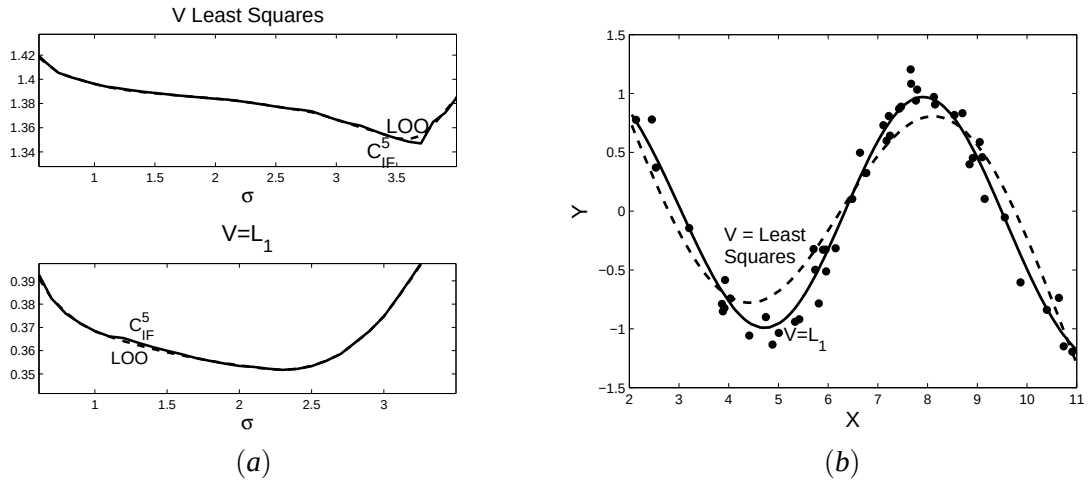


Figure 3: Data with outlier at $(4,5)$. The parameters $\lambda = 0.001$ and $\sigma = 2$ are fixed. Dashed: KBR with least squares loss function. Solid: KBR with Huber loss function ($b = 0.2$).

$\lambda = 0.001$ as a function of the bandwidth σ. We observe convergence from the training error towards the leave-one-out error as the number of terms included is increased. Unfortunately the convergence rate depends on the value of σ: convergence is quite slow at small values of σ. This is no surprise looking at (12). There we approximated the remainder term by a quantity depending on $(1-H_{i,i})^{-1}$. When σ is small, the diagonal elements of H become close to 1. In that case the deleted remainder term can indeed be quite large. Nevertheless, this approach can still be useful if some care is taken not to consider values of λ and σ that are too small. However, the criterion C_{IF}^5 from Definition 8 using the approximation in (12) is clearly superior. We see that the remainder term is now adequately estimated and a good approximation is obtained at any σ. The resulting curve is undistinguishable from the exact leave-one-out error. The mean absolute difference is $3.2\ 10^{-5}$, the maximal difference is $1.8\ 10^{-4}$. The CV approximation also yields a good result being indistinguishable from the exact leave-one-out error on the plot as well. The mean absolute difference is $4.1\ 10^{-4}$ and the maximal difference equals $1.8\ 10^{-3}$. Thus C_{IF}^5 is closer to the true leave-one-out error than CV, although the difference is irrelevant when it comes down to selecting a good σ.

Figure 2 also shows plots for the leave-one-out error and its various approximations at (b) $\lambda = 0.005$ as a function of σ, (c) $\sigma = 1$ as a function of λ, (d) $\sigma = 2$ as a function of λ. In these cases as well it is observed that the cutoff strategy yields decent results if a sufficient number of terms is taken into account and if one does not look at values of λ and σ that are extremely small. The best strategy is to take the remainder term into account using the criterion C_{IF}^k from Definition 8.

In Figure 3 we illustrate robustness. An (extreme) outlier was added at position $(4,5)$ (not visible on the plot). This outlier leads to a bad fit when LS-KBR is used with $\lambda = 0.001$ and $\sigma = 2$ (dashed line). When a Huber loss function is used with $b = 0.2$ a better fit is obtained that still nicely predicts the majority of observations. This behavior can be explained by Proposition 5. The least squares loss has an unbounded first derivative and thus the influence of outliers can be arbitrary large. The Huber loss has a bounded first derivative and thus the influence of outliers is bounded as



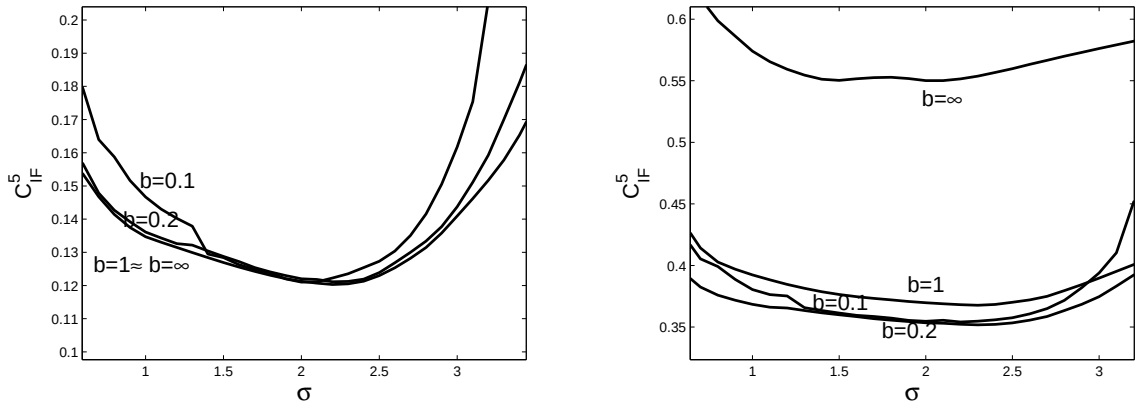
(a) (b)

Figure 4: (a) Optimization of σ at $\lambda = 0.001$. Upper: using least squares loss V in the model selection. Lower: using L_1 loss V in the model selection. For the estimation the loss function L is always the Huber loss with $b = 0.2$. (b) Resulting fits. Dashed line: $\sigma = 3.6$ (optimal choice using V least squares). Solid line: $\sigma = 2.3$ (optimal choice using L_1 loss for V.

well. However, note that in this example as well as in Proposition 5 the hyperparameters λ and σ are assumed to have fixed values. In practice one wants to choose these parameters in a data driven way. Figure $4(a)$ shows the optimization of σ at $\lambda = 0.001$ for KBR with L the Huber loss with $b = 0.2$. In the upper panel the least squares loss is used for V in the model selection criteria. Both exact leave-one-out and C^5_{IF} indicate that a value of $\sigma \approx 3.6$ should be optimal. This results in the dashed fit in Figure $4(b)$. In the lower panel of Figure 4 the L_1 loss is used for V in the model selection criteria. Both exact leave-one-out and C^5_{IF} indicate that a value of $\sigma \approx 2.3$ should be optimal. This results in the solid fit in Figure $4(b)$. We clearly see that, although in both cases a robust estimation procedure is used (Huber loss for L), the outlier can still be quite influential through the model selection. To obtain full protection against outliers, both the estimation and the model selection step require robustness, for example by selecting both L and V in a robust way.

Finally let us investigate the role of the parameter b used in the Huber loss function. We now use C^5_{IF} with V the L_1 loss. When we apply C^5_{IF} to the clean data without the outlier, we observe in Figure $5(a)$ that the choice of b does not play an important role. This is quite expected: since there are no outliers, there is no reason why least squares ($b = \infty$) would not perform well. On the contrary, if we use a small b such as $b = 0.1$ we get a slightly worse result. Again this is not a surprise, since with small b we will downweight a lot of points that are actually perfectly ok.

The same plot for the data containing the outlier yields a different view in Figure $5(b)$. The values of C^5_{IF} are much higher for least squares than for the Huber loss with smaller b. Thus it is automatically detected that a least squares loss is not the appropriate choice, which is a correct assessment since the outlier will have a large effect (cf. the dashed line in Figure 3). The criterion C^5_{IF} indicates a choice $b = 0.2$, which leads to a better result indeed (cf. the solid line in Figure 3)



(a) (b)

Figure 5: C^5_{IF} at $\lambda = 0.001$ as a function of σ for several values of b for (a) the clean data without the outlier, (b) the data with the outlier.

6.2 Other Examples

This part presents the results of a small simulation study. We consider some well known settings.

- Friedman 1 ($d = 10$): $y(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 1/2)^2 + 10x_4 + 5x_5 + \sum_{i=6}^{10} 0.x_i$. The covariates are generated uniformly in the hypercube in $\mathbb{R}^{10}$.

- Friedman 2 ($d = 4$): $y(x) = \frac{1}{3000}(x_1^2 + (x_2x_3 - (x_2x_4)^{-2}))^{1/2}$, with $0 < x_1 < 100$, $20 < x_2/(2\pi) < 280$, $0 < x_3 < 1$, $1 < x_4 < 11$.

- Friedman 3 ($d = 4$): $y(x) = \tan^{-1}(\frac{x_2x_3 - (x_2x_4)^{-2}}{x_1})$, with the same range for the covariates as in Friedman 2. For each of the Friedman data sets 100 observations were generated with Gaussian noise and 200 noise free test data were generated.

- Boston Housing Data from the UCI machine learning depository with 506 instances and 13 covariates. Each split 450 observations were used for training and the remaining 56 for testing.

- Ozone data from ftp://ftp.stat.berkeley.edu/pub/users/breiman/ with 202 instances and 12 covariates. Each split 150 observations were used for training and the remaining 52 for testing.

- Servo data from the UCI machine learning depository with 167 instances and 4 covariates. Each split 140 observations were used for training and the remaining 27 for testing.

For the real data sets (Boston, Ozone and Servo), new contaminated data set were constructed as well by adding large noise to 10 training points, making these 10 points outliers.

The hyperparameters λ and σ are optimized over the following grid of hyperparametervalues:

- $\lambda \in \{50, 10, 5, 3, 1, 0.8, 0.5, 0.3, 0.1, 0.08, 0.05, 0.01, 0.005\} \times 10^{-3}$.

- For each data set 500 distances were calculated between two randomly chosen observations. Let $d_{(i)}$ be the ith largest distance. Then the following grid of values for σ is considered: $\sigma \in \{\frac{1}{2}d_{(1)}, d_{(1)}, d_{(50)}, d_{(100)}, d_{(150)}, d_{(200)}, d_{(250)}, d_{(300)}, d_{(350)}, d_{(400)}, d_{(450)}, d_{(500)}, 2d_{(500)}\}$.

In each replicate the Mean Squared Error of the test data is computed. For every data set the average MSE over 20 replicates is shown in Table 1 (upper table). A two-sided paired Wilcoxon rank test is used to check statistical significance: values in italic are significantly different from the smallest value at significance level 0.05. If underlined significance holds even at significance level 10^{-4}. Standard errors are shown as well (lower table). First we consider the least squares loss for L with the criterion $C^5_{IF}(\lambda, \sigma, \infty)$ (Definition 8), with exact leave-one-out (17) and with CV (22). These are the first 3 columns in Table 1. We see that the difference between these 3 criteria is very small. This means that both CV and C^5_{IF} provide good approximations of the leave-one-out error.

Secondly, we considered each time the residuals of the least squares fit with optimal λ and σ according to $C^5_{IF}(\lambda, K, \infty)$. An estimate $\hat{\sigma}_{err}$ of the scale of the residuals is computed as the MAD of these residuals (18). Then we applied KBR with a Huber loss and parameter $b = 3\hat{\sigma}_{err}$. The resulting MSE with this loss and λ and σ minimizing $C^5_{IF}(\lambda, \sigma, 3\hat{\sigma}_{err})$ is given in column 4 in Table 1. Similar results are obtained for $b = 2\hat{\sigma}_{err}$ in column 5 and with $b = \hat{\sigma}_{err}$ in column 6. For the data sets without contamination we see that using a Huber loss instead of least squares gives similar results except for the Boston housing data, Friedman 1 and especially Friedman 2. For those data sets a small value of b is inappropriate. This might be explained by the relationship between the loss function and the error distribution. For a Gaussian error distribution least squares is often an optimal choice (cf. maximum likelihood theory). Since the errors in the Friedman data are explicitly generated as Gaussian, this might explain why least squares outperforms the Huber loss. For real data sets, the errors might not be exactly Gaussian, and thus other loss function can perform at least equally well as least squares. For the data sets containing the outliers the situation changes of course. Now least squares is not a good option because of its lack of robustness. Clearly the outliers have a large and bad effect on the quality of the predictions. This is not the case when the Huber loss function is chosen. Then the effect of the outliers is reduced. Choosing $b = 3\hat{\sigma}_{err}$ already leads to a large improvement. Decreasing b leads to even better results (note that the p-values are smaller than 10^{-4} for any significant pairwise comparison).

Finally we also consider optimizing b. We apply the algorithm outlined in Section 5.2. Corresponding MSE's are given in the last column of Table 1. For the Friedman 1 and Friedman 2 data sets for instance this procedure indeed detects that least squares is an appropriate loss function and automatically avoids choosing b too small. For the contaminated data sets the procedure detects that least squares is not appropriate and that changing to a Huber loss with a small b is beneficial, which is indeed a correct choice yielding smaller MSE's. In fact, only for the Friedman 2 data, the automatic choice of b is significantly worse than the optimal choice (p-value=0.03), whereas the benefits at the contaminated data are large (all p-values $< 10^{-4}$).

7. Conclusion

Heuristic links between the concept of the influence function and concepts as leave-one-out cross validation and stability were considered in Section 2, indicating some interesting applications of the influence function and the leave-one-out error in previous literature. New results include the calculation of higher order influence functions and a recursive relation between subsequent terms. It is shown that these theoretical results can be applied in practice to approximate the leave-one-out esti-

	$b=\infty$ (=LS)			$b=3\hat{\sigma}_{err}$	$b=2\hat{\sigma}_{err}$	$b=\hat{\sigma}_{err}$	($b=$ optimized)
	LOO	CV	C^5_{IF}	C^5_{IF}	C^5_{IF}	C^5_{IF}	C^5_{IF}
F1	1.63	1.63	1.63	1.66	*1.70*	*1.82*	1.67
F2	1.30	1.30	1.30	*1.42*	1.71	3.02	*1.39*
F3	2.42	2.42	2.42	2.42	2.42	2.37	2.38
B	10.58	10.58	10.58	10.82	11.30	*12.21*	10.79
O	13.91	13.92	13.91	13.76	13.73	13.91	13.94
S	0.40	0.40	0.40	0.43	0.41	0.41	0.40
B+o	37.54	37.54	37.54	14.60	13.73	12.68	12.78
O+o	78.78	78.78	78.77	21.20	18.85	16.74	16.74
S+o	1.60	1.60	1.60	0.61	0.54	0.46	0.46

	$b=\infty$ (=LS)			$b=3\hat{\sigma}_{err}$	$b=2\hat{\sigma}_{err}$	$b=\hat{\sigma}_{err}$	($b=$ optimized)
	LOO	CV	C^5_{IF}	C^5_{IF}	C^5_{IF}	C^5_{IF}	C^5_{IF}
F1	0.09	0.09	0.09	0.09	0.10	0.08	0.09
F2	0.14	0.14	0.15	0.16	0.20	0.36	0.15
F3	0.03	0.03	0.03	0.03	0.03	0.05	0.05
B	1.39	1.39	1.39	1.40	1.46	1.51	1.39
O	0.86	0.86	0.87	0.78	0.78	0.75	0.81
S	0.05	0.05	0.05	0.09	0.08	0.09	0.09
B+o	2.91	2.91	2.91	1.12	1.09	1.02	1.04
O+o	3.44	3.44	3.44	1.01	0.97	1.03	1.03
S+o	0.16	0.16	0.16	0.07	0.07	0.08	0.08

Table 1: Simulation results. Upper: Mean Squared Errors. Lower: standard errors. Friedman 1 (F1), Friedman 2 (F2), Friedman 3 (F3), Boston Housing (B), Ozone (O), Servo (S), Boston Housing with outliers (B+o), Ozone with outliers (O+o) and Servo with outliers (S+o). Italic values are significantly different from the smallest value in the row with p-value in between 0.05 and 0.001 using a paired Wilcoxon rank test; underlined values are significant with p-value $< 10^{-4}$.

mator. Experiments indicate that the quality of this approximation is quite good. The approximation is used in a model selection criterion to select the regularization and kernel parameters.

We discussed the importance of robustness in the model selection step. A specific procedure is suggested using an L_1 loss in the model selection criterion and a Huber loss in the estimation. Due to an iterative reweighting algorithm to compute such a Huber loss estimator and due to the fast approximation of the leave-one-out error, everything can be computed fast starting from the least squares framework. With an a priori choice of the parameter b in the Huber loss this leads to better robustness if b is chosen small enough. If b is chosen too small on the other hand this might result in worse predictions. However, this parameter can be selected in a data driven way as well. Experiments suggest that this often yields a good trade-off between the robustness of choosing a small b and the sometimes better predictive capacity of least squares.

Acknowledgments

JS acknowledges support from K.U. Leuven, GOA-Ambiorics, CoE EF/05/006, FWO G.0499.04, FWO G.0211.05, FWO G.0302.07, IUAP P5/22.

MH acknowledges support from FWO G.0499.04, the GOA/07/04-project of the Research Fund KULeuven, and the IAP research network nr. P6/03 of the Federal Science Policy, Belgium.

Appendix A.

Proof of Theorem 6

Let P be a distribution, $z \in \mathcal{X} \times \mathcal{Y}$ and $P_{\varepsilon,z} = (1-\varepsilon)P + \varepsilon\Delta_z$ with Δ_z the Dirac distribution in z. We start from the representer theorem of DeVito et al. (2004) (a generalization of (13)):

$$2\lambda f_{\lambda,K,P_{\varepsilon,z}} = \mathbb{E}_{P_{\varepsilon,z}}[L'(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)].$$

By definition of $P_{\varepsilon,z}$ and since $\mathbb{E}_{\Delta_z} g(X) = g(z)$ for any function g:

$$2\lambda f_{\lambda,K,P_{\varepsilon,z}} = (1-\varepsilon)\mathbb{E}_P[L'(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] + \varepsilon L'(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x).$$

Taking the first derivative on both sides with respect to ε yields

$$\begin{aligned}
2\lambda\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}} =& (1-\varepsilon)\mathbb{E}_P[-\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\
&- \mathbb{E}_P[L'(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] + L'(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x) \\
&- \varepsilon\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(z_x))L''(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x).
\end{aligned}$$

The second derivative equals

$$\begin{aligned}
2\lambda\frac{\partial}{\partial^2\varepsilon} f_{\lambda,K,P_{\varepsilon,z}} =& -\mathbb{E}_P[-\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\
&+ (1-\varepsilon)\mathbb{E}_P[-\frac{\partial}{\partial^2\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\
&+ (1-\varepsilon)\mathbb{E}_P[-\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(X)L'''(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))(-\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\
&- \mathbb{E}_P[L''(Y - f_{\lambda,K,P_{\varepsilon,z}}(X))(-\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\
&- \frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x) \\
&- \varepsilon\frac{\partial}{\partial^2\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x) \\
&- \varepsilon\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(z_x)L'''(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))(-\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x) \\
&- L''(z_y - f_{\lambda,K,P_{\varepsilon,z}}(z_x))\frac{\partial}{\partial\varepsilon} f_{\lambda,K,P_{\varepsilon,z}}(z_x)\Phi(z_x).
\end{aligned}$$

Simplifying yields

$$\begin{aligned}2\lambda\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}} =& 2\mathbb{E}_P[\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\ &-(1-\varepsilon)\mathbb{E}_P[\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\ &+(1-\varepsilon)\mathbb{E}_P[\left(\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)\right)^2 L'''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\ &-2\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x) \\ &-\varepsilon\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x) \\ &+\varepsilon\left(\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)\right)^2 L'''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x).\end{aligned} \tag{23}$$

Evaluating at $\varepsilon=0$ and bringing all terms containing $\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}$ to the left hand side of the equation yields

$$\begin{aligned}2\lambda\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}|_{\varepsilon=0}+\mathbb{E}_P[\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)|_{\varepsilon=0}L''(Y-f_{\lambda,K,P}(X))\Phi(X)] \\ =2\mathbb{E}_P[\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)|_{\varepsilon=0}L''(Y-f_{\lambda,K,P}(X))\Phi(X)] \\ +\mathbb{E}_P[\left(\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}|_{\varepsilon=0}(X)\right)^2 L'''(Y-f_{\lambda,K,P}(X)) \\ -2\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P}(z_x)|_{\varepsilon=0}L''(z_y-f_{\lambda,K,P}(z_x))\Phi(z_x).\end{aligned}$$

Since by definition $\frac{\partial}{\partial\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}|_{\varepsilon=0}$ is $IF(z;f_{\lambda,K},P)$ and $\frac{\partial}{\partial^2\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}|_{\varepsilon=0}$ is $IF_2(z;f_{\lambda,K},P)$ we have that

$$\begin{aligned}S(IF_2(z;f_{\lambda,K},P)) = 2\mathbb{E}_P[&IF(z;f_{\lambda,K},P)(X)L''(Y-f_{\lambda,K,P}(X))\Phi(X)] \\ &+\mathbb{E}_P[\big(IF(z;f_{\lambda,K},P)(X)\big)^2 L'''(Y-f_{\lambda,K,P}(X)) \\ &-2IF(z;f_{\lambda,K},P)(z_x)L''(z_y-f_{\lambda,K,P}(z_x))\Phi(z_x)\end{aligned}$$

with the operator S defined by $S: f\rightarrow\lambda f+\mathbb{E}_P L''(Y-f_{\lambda,K,P}(X))f(X)\Phi(X)$. Christmann and Steinwart (2007) prove that S is an invertible operator and thus Theorem 6 follows.

Proof of Theorem 7

First we proof the following for all $2\leq k\in\mathbb{N}$:

$$\begin{aligned}2\lambda\frac{\partial}{\partial^k\varepsilon}f_{\lambda,K}(P_{\varepsilon,z}) =& (1-\varepsilon)\mathbb{E}_P[-\frac{\partial}{\partial^k\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\ &+k\mathbb{E}_P[\frac{\partial}{\partial^{k-1}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)] \\ &-kL''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\frac{\partial}{\partial^{k-1}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)\Phi(z_x) \\ &-\varepsilon L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\frac{\partial}{\partial^k\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)\Phi(z_x).\end{aligned} \tag{24}$$

Note that for $k=2$ this immediately follows from (23). For general k we give a proof by induction. We assume that (24) holds for k and we then prove that it automatically holds for $k+1$ as well. Taking the derivatives of both sides in (24) we find

$$\begin{aligned}
\lambda\frac{\partial}{\partial^{k+1}\varepsilon}f_{\lambda,K}(P_{\varepsilon,z}) =&(1-\varepsilon)\mathbb{E}_P[-\frac{\partial}{\partial^{k+1}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)]\\
&-\mathbb{E}_P[-\frac{\partial}{\partial^{k}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)]\\
&+k\mathbb{E}_P[\frac{\partial}{\partial^{k}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)]\\
&-k\frac{\partial}{\partial^{k}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x)\\
&-\varepsilon\frac{\partial}{\partial^{k+1}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x)\\
&-\frac{\partial}{\partial^{k}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(z_x)L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x)
\end{aligned}$$

from which it follows that (24) holds for $k+1$ indeed. Evaluating this expression in $\varepsilon=0$ yields:

$$\begin{aligned}
&\lambda\frac{\partial}{\partial^{k+1}\varepsilon}f_{\lambda,K}(P_{\varepsilon,z})|_{\varepsilon=0}+\mathbb{E}_P[\frac{\partial}{\partial^{k+1}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)|_{\varepsilon=0}L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)]\\
&=(k+1)\mathbb{E}_P[\frac{\partial}{\partial^{k}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}(X)|_{\varepsilon=0}L''(Y-f_{\lambda,K,P_{\varepsilon,z}}(X))\Phi(X)]\\
&-(k+1)\frac{\partial}{\partial^{k}\varepsilon}f_{\lambda,K,P_{\varepsilon,z}}|_{\varepsilon=0}(z_x)L''(z_y-f_{\lambda,K,P_{\varepsilon,z}}(z_x))\Phi(z_x).
\end{aligned}$$

Thus

$$\begin{aligned}
S(IF_{k+1}(z;f_{\lambda,K},P)) = (k+1)\Big(&\mathbb{E}_P[IF_k(z;f_{\lambda,K},P)(X)L''(Y-f_{\lambda,K}(X))\Phi(X)]\\
&-[IF_k(z;f_{\lambda,K},P)(z_x)L''(z_y-f_{\lambda,K}(z_x))\Phi(z_x)]\Big).
\end{aligned}$$

Since S is an invertible operator the result in Theorem 7 follows.

References

D.D. Boos and R.J. Serfling. A note on differentials and the CLT and LIL for statistical functions. *Annals of Statistics*, 8:618–624, 1980.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2001.

A. Christmann and I. Steinwart. Consistency and robustness of kernel based regression. *Bernoulli*, 13:799–819, 2007.

A. Christmann and I. Steinwart. On robust properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research*, 5:1007–1034, 2004.

M. Debruyne, A. Christmann, M. Hubert, and J.A.K. Suykens. Robustness and stability of reweighted kernel based regression. Technical report TR 06-09, K.U. Leuven, available at http://wis.kuleuven.be/stat/robust, 2006.

E. DeVito, L. Rosasco, A. Caponnetto, M. Piana, and A. Verri. Some properties of regularized kernel methods. *Journal of Machine Learning Research*, 5:1363–1390, 2004.

T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

L.T. Fernholz. *Von Mises Calculus for Statistical Functionals*. Lecture Notes in statistics 19, Springer, New York, 1983.

F.R. Hampel, E.M. Ronchetti, P.J. Rousseeuw, and W.A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, New York, 1986.

P.J. Huber. *Robust Statistics*. Wiley, New York, 1981.

S. Kutin and P. Niyogi. Almost everywhere algorithmic stability and generalization error. In A. Daruich and N. Friedman, editors, *Proceedings of Uncertainty in AI*. Morgan Kaufmann, Edmonton, 2002.

T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

J.A.K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle. Weighted least squares support vector machines : Robustness and sparse approximation. *Neurocomputing*, 48:85–105, 2002a.

J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002b.

A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill Posed Problems*. W.H. Winston, Washington D.C., 1977.

G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 1990.

Journal of Machine Learning Research 9 (2008) 2401-2429 Submitted 7/07; Revised 5/08; Published 10/08

Non-Parametric Modeling of Partially Ranked Data

Guy Lebanon LEBANON@CC.GATECH.EDU
College of Computing
Georgia Institute of Technology
Atlanta, GA

Yi Mao YMAO@ECE.PURDUE.EDU
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN

Editor: Tommi Jaakkola

Abstract

Statistical models on full and partial rankings of n items are often of limited practical use for large n due to computational consideration. We explore the use of non-parametric models for partially ranked data and derive computationally efficient procedures for their use for large n. The derivations are largely possible through combinatorial and algebraic manipulations based on the lattice of partial rankings. A bias-variance analysis and an experimental study demonstrate the applicability of the proposed method.

Keywords: ranked data, partially ordered sets, kernel smoothing

1. Introduction

Rankers such as people, search engines, and classifiers, output full or partial rankings representing preference relations over n items or alternatives. For example in the case of $m = 6$ rankers issuing full or partial preferences over $n = 3$ items a possible data set is

$$3 \prec 1 \prec 2, \quad 3 \prec 2 \prec 1, \quad 1 \prec 3 \prec 2, \quad 1 \prec \{2,3\}, \quad 3 \prec \{1,2\}, \quad \{2,3\} \prec 1. \tag{1}$$

The first three expressions in (1) correspond to full rankings while the last three expressions correspond to partial rankings (the numbers correspond to items and the $\prec$ symbol corresponds to a preference relation). While it is likely that some rankings will contradict others, it is natural to assume that the data in (1) was sampled iid from some distribution p over rankings. The goal of this paper is to study non-parametric methods for the estimation of p based on data sets such as (1) in the case of large n.

Often, ranked data is not inherently associated with numeric score information. In other cases, numeric scores are available but are un-calibrated and cannot be compared to each other. For example, the assignment of numeric scores by people to items or alternatives is un-calibrated as each person has his or her own notion of what constitutes a certain numeric score. On the other hand, a preference of one item or alternative over another reflects a binary choice that is directly comparable across rankers. Thus, even in cases where numeric scores exist, modeling the scoreless preferences may achieve higher modeling accuracy.

Despite this motivating observation, modeling ranked data is less popular than modeling the existing numeric scores, or even made-up numeric scores in case the true scores are unavailable (such is the case with the frequently used Borda count). The main reason for this is that rankings over a large number of items n reside in an extremely large discrete space whose modeling often requires intractable computation.

Previous attempts at modeling ranked data have been mostly parametric and often designed to work with fully ranked data (Marden, 1996). Non-parametric modeling of fully ranked data has been recently addressed in the context of multi-object tracking (Kondor et al., 2007; Huang et al., 2008). They focus on maintaining and updating a distribution over permutations by a low frequency approximation of the distribution. Such an approximation results from a spectral decomposition of functions on the symmetric group on n items (Diaconis, 1988) and is essential for efficient probabilistic inference.

Most of aforementioned approaches are unsuitable for modeling partial rankings for medium and large n due to the computational difficulties of handling a probability space of size $n!$. The few possible exceptions (Critchlow, 1985; Marden, 1996) are usually more ad-hoc and do not correspond to an underlying permutation model making them ill suited to handle partial rankings of different types. In fact, most of the ranked data analyzed in the literature are limited to $n \leq 15$ and usually even $n \leq 5$ such as the popular APA election data set.

On the other hand, there has been a recent increase in data sets containing partial or full rankings for large n. Examples include (i) web-search data such as TREC[1] where n may be thought of as corresponding to the number of web-pages or approaching $+\infty$, (ii) movie review data sets such as the Netflix data set[2] where $n \approx 18000$ and MovieLens[3] where $n = 1682$, and (iii) multi-label text document data sets such as OHSUMED[4] where $n = 4904$ and Reuters RCV1[5] where $n = 103$. More details on how these data sets correspond to partial rankings may be found in Section 2.

These data sets and others lead to a growing number of somewhat ad-hoc but computationally efficient rank aggregation techniques. The techniques, developed primarily within the computer science community, are often non-probabilistic and output a single ranking summarizing the data. Unfortunately, such a summary ranking, while being useful, does not provide the data analysis capabilities offered by a full probabilistic model.

The main contribution of this paper is in proposing and studying a non-parametric estimator based on kernel smoothing for the estimation of the population distribution p. Some properties of the estimator are listed below. We are not aware of any other non-trivial estimator of p that satisfies these requirements, in particular for the case of large n.

(1) Estimate p based on full as well as partial rankings.

(2) The resulting estimate $\hat{p}$ should assign probabilities to full and partial rankings in a coherent and contradiction-free manner (described in Section 4).

(3) Estimate p based on partial rankings of different types (defined in Section 2).

1. TREC can be found at `http://trec.nist.gov/`.
2. Netflix can be found at `http://www.netflixprize.com/`.
3. MovieLens can be found at `http://www.grouplens.org/node/12/`.
4. OHSUMED can be found at `http://trec.nist.gov/data/t9_filtering/`.
5. RCV1 can be found at `http://trec.nist.gov/data/reuters/reuters.html/`.

(4) Statistical consistency $\hat{p} \xrightarrow{p} p$ as both the number of samples m and the number of items n grow to infinity.

(5) Statistical accuracy of $\hat{p}$ can be slow for fully ranked data but should be accelerated when restricted to simpler partial rankings.

(6) Obtaining the estimate $\hat{p}$ and using it to compute probabilities $\hat{p}(A)$ of partial rankings should be computationally feasible, even for large n.

All 6 properties above are crucial in the large n scenario: it is often impossible for rankers to specify full rankings over a very large number of items making the use of partial rankings a necessity. Different rankers may choose to output partial rankings of different types, for example, one ranker can output $3 \prec \{1,2\}$ (3 is preferred to both 1 and 2) and another ranker can output $\{1,3\} \prec 2$ (both 1 and 3 are preferred to 2). By considering the asymptotics $n \to \infty$ in addition to $m \to \infty$ (m being the number of samples) we provide a more realistic analysis for a large (and potentially growing) number of items. Computational feasibility is a major concern since most ranking models are incapable of modeling the data sets mentioned above due to their large n.

We continue next by reviewing basic concepts concerning partially ranked data and the Mallows model, and then proceed to define our non-parametric estimator. We conclude by demonstrating computational efficiency, statistical properties, and some experiments.

2. Permutations and Cosets

We begin by reviewing some basic concepts concerning permutations, with some of the notations and definitions borrowed from Critchlow (1985).

A permutation π is a bijective function $\pi : \{1,\ldots,n\} \to \{1,\ldots,n\}$ associating with each item $i \in \{1,\ldots,n\}$ a rank $\pi(i) \in \{1,\ldots,n\}$. In other words, $\pi(i)$ denotes the rank given to item i and $\pi^{-1}(i)$ denotes the item assigned to rank i. We denote a permutation π using the following vertical bar notation $\pi^{-1}(1)|\pi^{-1}(2)|\cdots|\pi^{-1}(n)$. For example, the permutation $\pi(1) = 2, \pi(2) = 3, \pi(3) = 1$ would be denoted as $3|1|2$. In this notation the numbers correspond to items and the locations of the items in their corresponding compartments correspond to their ranks. The collection of all permutations of n items forms the non-Abelian symmetric group of order n, denoted by $\mathfrak{S}_n$, using function composition as the group operation $\pi\sigma = \pi \circ \sigma$. We denote the identity permutation by e.

The concept of inversions and the result below will be of great use later on.

Definition 1 *The inversion set of a permutation π is the set of pairs*

$$U(\pi) \stackrel{\text{def}}{=} \{(i,j) : i < j,\ \pi(i) > \pi(j)\} \subset \{1,\ldots,n\} \times \{1,\ldots,n\}$$

whose cardinality is denoted by $i(\pi) \stackrel{\text{def}}{=} |U(\pi)|$.

For example, $i(e) = |\emptyset| = 0$, and $i(3|2|1|4) = |\{(1,2),(1,3),(2,3)\}| = 3$.

Proposition 1 (for example, Stanley, 2000) *The map $\pi \mapsto U(\pi)$ is a bijection.*

When n is large, the enormous number of permutations raises difficulties in using the symmetric group for modeling rankings. A reasonable solution is achieved by considering partial rankings which correspond to cosets of the symmetric group. For example, the subgroup of $\mathfrak{S}_n$ consisting of

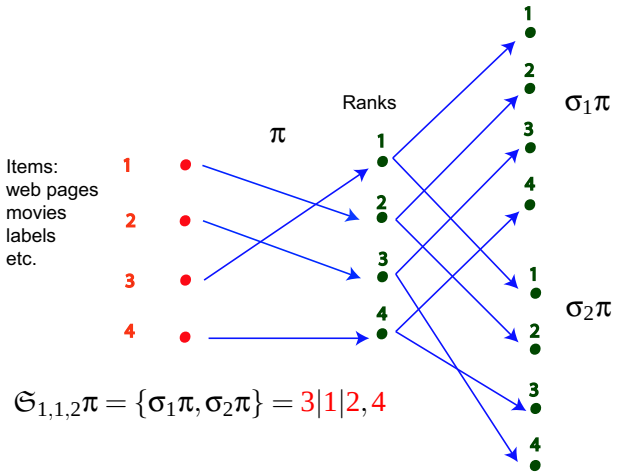


Figure 1: A partial ranking corresponds to a coset or a set or permutations

all permutations that fix the top k positions is denoted $\mathfrak{S}_{1,\ldots,1,n-k} = \{\pi \in \mathfrak{S}_n : \pi(i) = i,\ i = 1,\ldots,k\}$. The right coset $\mathfrak{S}_{1,\ldots,1,n-k}\pi = \{\sigma\pi : \sigma \in \mathfrak{S}_{1,\ldots,1,n-k}\}$ is the set of permutations consistent with the ordering of π on the k top-ranked items. It may thus be interpreted as a partial ranking of the top k items, that does not contain any information concerning the relative ranking of the bottom $n-k$ items. The set of all such partial rankings forms the quotient space $\mathfrak{S}_n/\mathfrak{S}_{1,\ldots,1,n-k}$. Figure 1 illustrates the identification of a coset as a partial ranking of the top 2 out of 4 items.

We generalize the above relationship between partial rankings and cosets through the following definition of a composition.

Definition 2 *A composition of n is a sequence $\gamma = (\gamma_1,\ldots,\gamma_r)$ of positive integers whose sum is n.*

Note that in contrast to a partition, in a composition the order of the integers matters. A composition $\gamma = (\gamma_1,\ldots,\gamma_r)$ corresponds to a partial ranking with γ_1 items in the first position, γ_2 items in the second position and so on. For such a partial ranking it is known that the first set of γ_1 items are to be ranked before the second set of γ_2 items etc., but no further information is conveyed about the orderings within each set. The partial ranking introduced earlier $\mathfrak{S}_{1,\ldots,1,n-k}\pi$ of the top k items is a special case corresponding to $\gamma = (1,\ldots,1,n-k)$.

More formally, let $N_1 = \{1,\ldots,\gamma_1\}, N_2 = \{\gamma_1+1,\ldots,\gamma_1+\gamma_2\},\cdots,N_r = \{\gamma_1+\cdots+\gamma_{r-1}+1,\ldots,n\}$. The subgroup $\mathfrak{S}_\gamma$ is defined as the set of all permutations $\pi \in \mathfrak{S}_n$ for which the following set equalities hold (the two sets on the left hand side and right hand side of the equality contain the same elements)

$$\pi(N_i) = N_i \qquad i = 1,\ldots,r.$$

In other words, the subgroup $\mathfrak{S}_\gamma$ contains permutations that only permute within each set N_i. It can be shown that the subgroup $\mathfrak{S}_\gamma$ is isomorphic to the product of subgroups $\mathfrak{S}_{\gamma_1} \times \cdots \times \mathfrak{S}_{\gamma_r}$ and is sometimes described by that product for notational purposes. A partial ranking of type γ is equivalent to a coset $\mathfrak{S}_\gamma\pi = \{\sigma\pi : \sigma \in \mathfrak{S}_\gamma\}$ and the set of such partial rankings forms the quotient space $\mathfrak{S}_n/\mathfrak{S}_\gamma$.

The vertical bar notation described above for permutations is particularly convenient for denoting partial rankings. We list items $1,\ldots,n$ separated by vertical bars, indicating that items on the left side of each vertical bar are preferred to (ranked higher than) items on the right side of the bar. On the other hand, there is no knowledge concerning the preference of items that are not separated by one or more vertical bars. For example, the partial ranking displayed in Figure 1 is denoted by $3|1|2,4$. The ordering of items not separated by a vertical line is meaningless, and for consistency we use the conventional ordering, for example, $1|2,3|4$ rather than the equivalent $1|3,2|4$.

The set of all partial rankings

$$\mathfrak{W}_n \stackrel{\text{def}}{=} \{\mathfrak{S}_\gamma \pi : \pi \in \mathfrak{S}_n, \forall \gamma\} \tag{2}$$

which includes the set of full rankings $\mathfrak{S}_n$, is a subset of all possible partial orders on $\{1,\ldots,n\}$. While the formalism of partial rankings in $\mathfrak{W}_n$ cannot realize all partial orderings, it is sufficiently powerful to include many useful and naturally occurring orderings as special cases. Furthermore, as demonstrated in later sections, it enables simplification of the otherwise overwhelming computational difficulty. Special cases of particular interest are the following partial rankings

- $\pi \in \mathfrak{S}_n$ corresponds to a permutation or a full ordering, for example, $3|2|4|1$.
- $\mathfrak{S}_{1,n-1}\pi$, for example, $3|1,2,4$, corresponds to selection of the top alternative. An example for such a ranking is a classification of x by a response variable $y \in \mathcal{Y} = \{1,\ldots,n\}$.
- $\mathfrak{S}_{1,\ldots,1,n-k}\pi$, for example, $1|3|2,4$, corresponds to full ordering of the top k items. An example for such a ranking is a ranked list of the top k webpages output by search engines in response to a query.
- $\mathfrak{S}_{k,n-k}\pi$, for example, $1,2,4|3,5$, corresponds to a more preferred and a less preferred dichotomy. Alternatively the dichotomy can be interpreted as right and wrong or relevant and irrelevant. An example for such a ranking is classification of alternatives into desirable and undesirable.
- $\mathfrak{S}_{1,\ldots,1,n-k-t,1,\ldots,1}\pi$, for example, $5|1|2,4,7|6|3|8$, corresponds to full ordering of the top k and the bottom t items. An example for such a ranking is a list of the safest and the most dangerous U.S. cities.[6]
- $\mathfrak{S}_{k,n-k-t,t}\pi$, for example, $1,5|2,4,7|3,6,8$, corresponds to a trichotomy of items. An example for such a ranking is selection of preferred and non-preferred items from a list.

Traditionally, data from each one of the special cases above was modeled using different tools and was considered fundamentally different. That problem was aggravated as different special cases were usually handled by different communities (statistics, computer science, information retrieval). As a first step towards presenting a unified framework for modeling partially ranked data, Lebanon and Lafferty (2003) demonstrated equivalence between several popular conditional models. We continue along this line and present in this paper a non-parametric framework capable of efficiently modeling a large variety of partially ranked data.

In constructing a statistical model on permutations or cosets, it is essential to relate one permutation to another. We do this using a distance function on permutations $d : \mathfrak{S}_n \times \mathfrak{S}_n \to \mathbb{R}$ that

6. List can be found at `http://www.infoplease.com/ipa/A0921299.html`.

satisfies the usual metric function properties, and in addition is invariant under right action of the symmetric group (Critchlow, 1985)

$$d(\pi,\sigma) = d(\pi\tau,\sigma\tau) \quad \forall\, \pi,\sigma,\tau \in \mathfrak{S}_n. \tag{3}$$

The invariance requirement (3) ensures that the distance does not change if the labeling of the items $\{1,\ldots,n\}$ (which is assumed to be arbitrary) is permuted.

There have been many propositions for such right-invariant distance functions, the most popular of them being Kendall's tau (Kendall, 1938)

$$d(\pi,\sigma) = \sum_{i=1}^{n-1}\sum_{l>i} I(\pi\sigma^{-1}(i) - \pi\sigma^{-1}(l)) \tag{4}$$

where $I(x) = 1$ for $x > 0$ and $I(x) = 0$ otherwise. Kendall's tau $d(\pi,\sigma)$ (4) measures the number of pairs of items for which π and σ have opposing orderings (also called disconcordant pairs). An equivalent definition for Kendall's tau is the minimum number of adjacent transpositions needed to bring π^{-1} to σ^{-1} (adjacent transposition flips a pair of items having adjacent ranks). By right invariance, $d(\pi,\sigma) = d(\pi\sigma^{-1}, e)$ which, for Kendall's tau equals the number of inversions $i(\pi\sigma^{-1})$. This is an important observation that will allow us to simplify many expressions concerning Kendall's tau using the combinatorial properties of inversions.

Kendall's tau $d(\pi,\sigma), \pi,\sigma \in \mathfrak{S}_n$ takes values between 0 for $\pi = \sigma$ and $n(n-1)/2$. It is sometimes desirable to consider the normalized Kendall's tau

$$d_n(\pi,\sigma) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1}\sum_{l>i} I(\pi\sigma^{-1}(i) - \pi\sigma^{-1}(l)) \tag{5}$$

whose range is $[0,1]$ and consequentially may be compared across different values of n.

3. The Mallows Model and its Extension to Partial Rankings

The Mallows model (Mallows, 1957) is a location-scale model on permutations based on Kendall's tau distance

$$p_\kappa(\pi) = \exp\left(-c\, d(\pi,\kappa) - \log\psi(c)\right) \qquad \pi,\kappa \in \mathfrak{S}_n \quad c \in \mathbb{R}_+.$$

The normalization term $\psi(c) = \sum_{\pi\in\mathfrak{S}_n} \exp(-c\, d(\pi,\kappa))$ does not depend on the location parameter κ and has the closed form

$$\begin{aligned}\psi(c) &= \sum_{\pi\in\mathfrak{S}_n} e^{-c\, d(\pi,\kappa)} \\ &= (1+e^{-c})(1+e^{-c}+e^{-2c})\cdots(1+e^{-c}+\cdots+e^{-(n-1)c}) \\ &= \prod_{j=1}^{n} \frac{1-e^{-jc}}{1-e^{-c}}\end{aligned} \tag{6}$$

as shown by the fact that $d(\pi,\kappa) = i(\pi\kappa^{-1})$ and the following proposition.

Proposition 2 (for example, Stanley, 2000) *For $q > 0$, $\sum_{\pi\in\mathfrak{S}_n} q^{i(\pi)} = \prod_{j=1}^{n-1}\sum_{k=0}^{j} q^k$.*

Proof Due to the bijection between permutations and sets of inversions expressed in Proposition 1

$$\sum_{\pi\in\mathfrak{S}_n} q^{i(\pi)} = \sum_{a_1=0}^{n-1}\sum_{a_2=0}^{n-2}\cdots\sum_{a_n=0}^{0} q^{a_1+\ldots+a_n} = \left(\sum_{a_1=0}^{n-1} q^{a_1}\right)\left(\sum_{a_2=0}^{n-2} q^{a_2}\right)\cdots\left(\sum_{a_n=0}^{0} q^{a_n}\right)$$
$$= (1+q+\cdots+q^{n-1})\cdots(1+q+q^2)(1+q)1.$$

■

The Mallows model has been motivated on axiomatic grounds by Mallows and has been a major focus of statistical modeling on permutations. Various extensions of the Mallows model may be found in Fligner and Verducci (1986, 1988, 1993). One particular extension to partial rankings is to consider a partial ranking as censored data equivalent to the set of permutations in its related coset. In other words, we define the probability the model assigns to the partial ranking $\mathfrak{S}_\gamma\pi$ by

$$\sum_{\tau\in\mathfrak{S}_\gamma\pi} p_\kappa(\tau) = \psi^{-1}(c)\sum_{\tau\in\mathfrak{S}_\gamma\pi}\exp\left(-c\,d(\tau,\kappa)\right). \tag{7}$$

Fligner and Verducci (1986) showed that in the case of $\gamma = (1,\ldots,1,n-k)$ the summation in (7) has a simple closed form. However, the apparent absence of a closed form formula for more general partial rankings prevented the widespread use of Equation 7 for large n and encouraged more ad-hoc and heuristic models (Critchlow, 1985; Marden, 1996). Section 7 describes an efficient computational procedure for computing (7) for more general partial ranking types γ.

4. The Ranking Lattice

Partial rankings $\mathfrak{S}_\gamma\pi$ relate to each other in a natural way by expressing more general, more specific or inconsistent ordering. We define below the concepts of partially ordered sets and lattices and then relate them to partial rankings by considering the set of partial rankings $\mathfrak{W}_n$ as a lattice. Some of the definitions below are taken from Stanley (2000), where a thorough introduction to posets can be found.

Definition 3 *A partially ordered set or poset $(Q,\preceq)$, is a set Q endowed with a binary relation $\preceq$ satisfying $\forall x,y,z\in Q$ (i) reflexibility: $x\preceq x$, (ii) anti-symmetry: $x\preceq y$ and $y\preceq x\Rightarrow x=y$, and (iii) transitivity: $x\preceq y$ and $y\preceq z\Rightarrow x\preceq z$.*

We write $x\prec y$ when $x\preceq y$ and $x\neq y$. We say that y covers x when $x\prec y$ and there is no $z\in Q$ such that $x\prec z\prec y$. A finite poset is completely described by its covering relation. The planar Hasse diagram of $(Q,\preceq)$ is the graph connecting the elements of Q as nodes using edges that correspond to the covering relation. An additional requirement is that if y covers x then y is drawn higher than x. Two elements x,y are comparable if $x\preceq y$ or $y\preceq x$ and otherwise are incomparable. The set of partial rankings $\mathfrak{W}_n$ defined in (2) is naturally endowed with the partial order of ranking refinement, that is, $\pi\prec\sigma$ if π refines σ or alternatively if we can get from π to σ by dropping vertical lines (Lebanon and Lafferty, 2003). Figure 2 shows the Hasse diagram of $\mathfrak{W}_3$ and a partial Hasse diagram of $\mathfrak{W}_4$.

An interesting visualization of Kendall's tau distance $d(\pi,\sigma),\pi,\sigma\in\mathfrak{S}_n$ in terms of the Hasse diagram is that it is the minimum number of up and down moves needed to get from π to σ on the

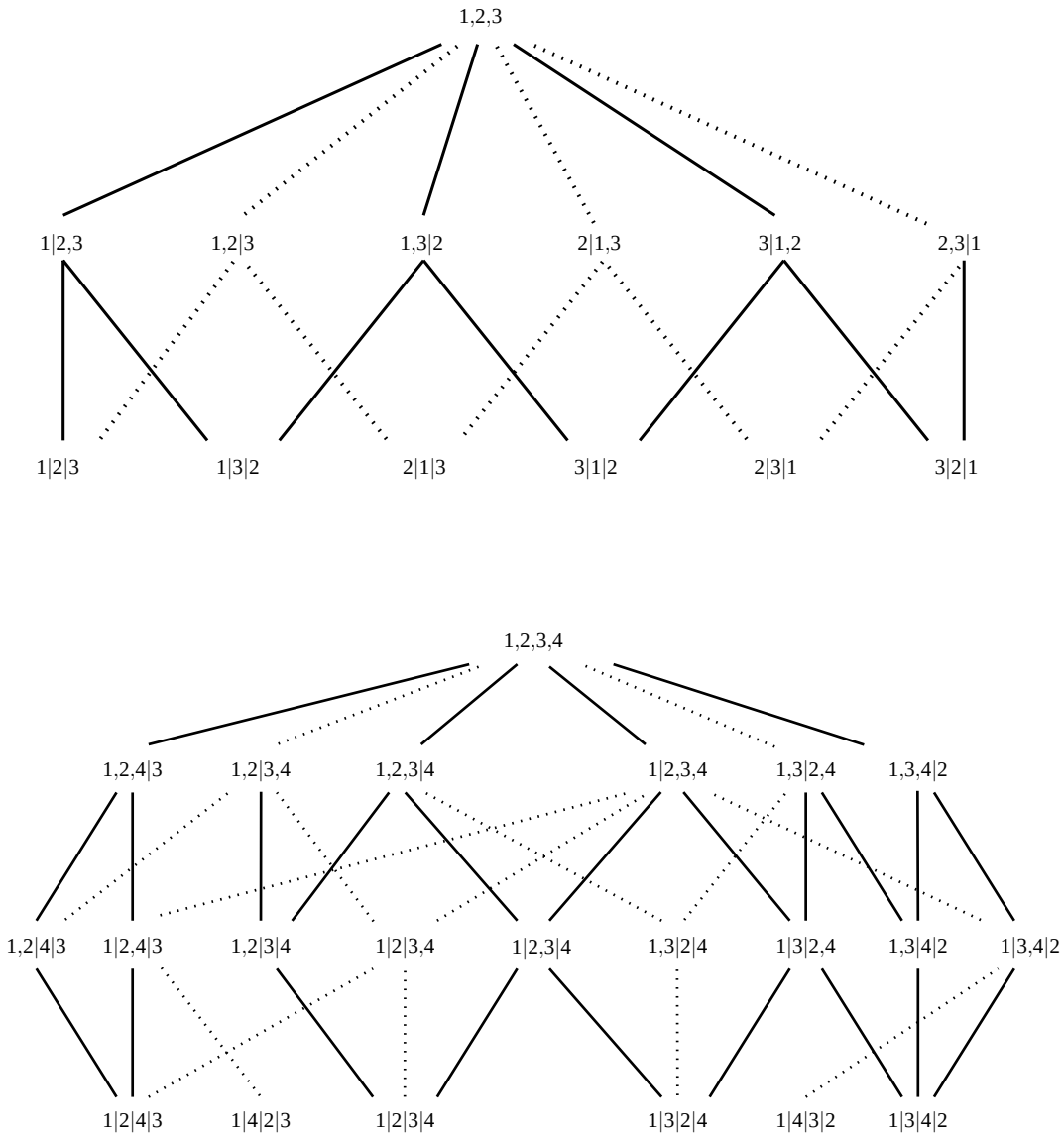


Figure 2: The Hasse diagram of $\mathfrak{W}_3$ (top) and a partial Hasse diagram of $\mathfrak{W}_4$ (bottom). Some of the lines are dotted for 3D visualization purposes (think 3D).

Hasse diagram. For example, in Figure 2 (top) we have $d(1|2|3,3|2|1) = 3$ realized by the three up-down moves along the shortest path

$$1|2|3\ (\nearrow 1,2|3 \searrow 2|1|3)\ (\nearrow 2|1,3 \searrow 2|3|1)\ (\nearrow 2,3|1 \searrow 3|2|1).$$

A lower bound z of two elements in a poset x,y satisfies $z \preceq x$ and $z \preceq y$. The greatest lower bound of x,y or infimum is a lower bound of x,y that is greater than or equal to any other lower bound of x,y. Infimum, and the analogous concept of supremum are denoted by $x \wedge y$ and $x \vee y$ or

$\bigwedge\{x_1,\ldots,x_k\}$ and $\bigvee\{x_1,\ldots,x_k\}$ respectively. Two elements $x,y \in \mathfrak{W}_n$ are said to be consistent if there exists a lower bound in $\mathfrak{W}_n$. Note that consistency is a weaker relation than comparability. For example, $1|2,3|4$ and $1,2|3,4$ are consistent but incomparable while $1|2,3|4$ and $2|1,3|4$ are both inconsistent and incomparable. Using the vertical bar notation, two elements are inconsistent iff there exist two items i,j that appear on opposing sides of a vertical bar in x and y, that is, $x = \cdots i|j \cdots$ while $y = \cdots j|i \cdots$. A poset for which $\wedge$ and $\vee$ always exist is called a lattice. Lattices satisfy many useful combinatorial properties - one of which is that they are completely described by the $\wedge$ and $\vee$ operations. In fact lattices are often defined by the supremum and infimum relation, rather than by the partial order. While the ranking poset is not a lattice, it may be turned into one by augmenting it with a minimum element $\hat{0}$.

Proposition 3 *The union $\tilde{\mathfrak{W}}_n \stackrel{\text{def}}{=} \mathfrak{W}_n \cup \{\hat{0}\}$ of the ranking poset and a minimum element is a lattice.*

Proof Since $\tilde{\mathfrak{W}}_n$ is finite, it is enough to show existence of $\wedge,\vee$ for pairs of elements (Stanley, 2000). We begin by showing existence of $x \wedge y$. If x,y are inconsistent, there is no lower bound in $\mathfrak{W}_n$ and therefore the unique lower bound $\hat{0}$ is also the infimum $x \wedge y$. If x,y are consistent, their infimum may be obtained as follows. Since x and y are consistent, we do not have a pair of items i,j appearing as $i|j$ in x and $j|i$ in y. As a result we can form a lower bound z to x,y by starting with a list of numbers and adding the vertical bars that are in either x or y, for example for $x = 3|1,2,5|4$ and $y = 3|2|1,4,5$ we have $z = 3|2|1,5|4$. The resulting $z \in \mathfrak{W}_n$, is smaller than x and y since by construction it contains all preferences (encoded by vertical bars) in x and y. It remains to show that for every other lower bound z' of x and y we have $z' \preceq z$. If z' is comparable to z, $z' \preceq z$ since removing any vertical bar from z results in an element that is not a lower bound. If z' is not comparable to z, then both z,z' contain the vertical bars in x and vertical bars in y possibly with some additional ones. By construction z contains only the essential vertical bars to make it a lower bound and hence $z' \prec z$, contradicting the assumption that z,z' are non-comparable.

By Proposition 3.3.1 of Stanley (2000) a poset for which an infimum is always defined and that has a supremum element is necessarily a lattice. Since we just proved that $\wedge$ always exists for $\tilde{\mathfrak{W}}_n$ and $1,\ldots,n = \bigvee \tilde{\mathfrak{W}}_n$, the proof is complete. ■

5. Probabilistic Models on the Ranking Lattice

The ranking lattice is a convenient framework to define and study probabilistic models on partial rankings. Given a probability model p on $\mathfrak{S}_n$, we define the functions $h,g : \tilde{\mathfrak{W}}_n \to [0,1]$

$$h(\alpha) = \begin{cases} p(\alpha) & \alpha \in \mathfrak{S}_n \\ 0 & \alpha \in \tilde{\mathfrak{W}}_n \setminus \mathfrak{S}_n \end{cases}$$
$$g(\alpha) = \sum_{\beta \in \tilde{\mathfrak{W}}_n : \beta \preceq \alpha} h(\beta). \tag{8}$$

Interpreting partial rankings $\mathfrak{S}_\gamma\pi \in \tilde{\mathfrak{W}}_n$ as the disjoint union of the events defined by the coset $\mathfrak{S}_\gamma\pi$ we have that

$$g(\mathfrak{S}_\gamma\pi) = \sum_{\tau \in \mathfrak{S}_\gamma\pi} p(\tau) \tag{9}$$

may be interpreted as the probability under p of the disjoint union $\mathfrak{S}_\gamma\pi$ of permutations. We refer to the function g as the partial ranking or lattice version of p. The motivation for defining g through h and not directly through p is that Equation (8) may be described and computed by the mechanism of Möbius inversion on lattices. More specifically, the Möbius inversion on lattices states that for two arbitrary real-valued functions on a lattice $h, g : \tilde{\mathfrak{W}}_n \to [0,1]$ we have

$$g(\tau) = \sum_{\tau' \preceq \tau} h(\tau') \quad \text{iff} \quad h(\tau) = \sum_{\tau' \preceq \tau} g(\tau')\mu(\tau', \tau) \qquad \tau, \tau \in \tilde{\mathfrak{W}}_n$$

where $\mu : \tilde{\mathfrak{W}}_n \times \tilde{\mathfrak{W}}_n \to \mathbb{R}$ is the Möbius function of the lattice $\tilde{\mathfrak{W}}_n$. In a certain sense this relationship between p and g generalizes the relationship between a probability mass function and the corresponding cdf. More details on Möbius functions and Möbius inversion on lattices and their computation may be found in Stanley (2000).

The function g is defined on the entire lattice, but when restricted to partial rankings of the same type $G = \{\mathfrak{S}_\gamma\pi : \pi \in \mathfrak{S}_n\} \subset \tilde{\mathfrak{W}}_n$, constitutes a normalized probability distribution on G. Estimating and examining a restriction of g to a subset $H \subset \tilde{\mathfrak{W}}_n$ (note that in general H may include more than one coset space G) rather than the function p is particularly convenient in cases of large n since H is often much smaller than the unwieldy $\mathfrak{S}_n$. In such cases it is tempting to specify the function g directly on H without referring to an underlying permutation model. However, doing so may lead to probabilistic contradictions such as $g(\mathfrak{S}_\gamma\pi) < g(\mathfrak{S}_\lambda\sigma)$ for $\mathfrak{S}_\lambda\sigma \subset \mathfrak{S}_\gamma\pi$. To avoid these and other probabilistic contradictions, g needs to satisfy a set of linear constraints equivalent to the existence of an underlying permutation model. Figure 3 illustrates this problem for partial rankings with the same (left) and different (right) number of vertical bars. A simple way to avoid such contradictions and satisfy the constraints is to define g indirectly in terms of a permutation model p as in (9). Applied to the context of statistical estimation, we define the estimator $\hat{g}$ in terms of an estimator $\hat{p}$ of the underlying permutation model p.

In addition to this construction which logically occurs after obtaining the estimator $\hat{p}$, we also need to consider how to use partially ranked data in the process of obtaining the estimator $\hat{p}$. Fully ranked data is often not available for large n since it is difficult for rankers (both human and others) to express with confidence full orderings over many items. Instead, the inference needs to be conducted based on a set of partial rankings

$$D = \{\mathfrak{S}_{\gamma_i}\pi_i : i = 1, \ldots, m\}. \tag{10}$$

A general way of using D in (10) to estimate $\hat{p}$ both parametrically and non-parametrically is to consider partially ranked data as censored or missing data. In other words, in the process of estimating $\hat{p}$, the data $\mathfrak{S}_\gamma\pi$ is considered as a single unknown permutation $\sigma \in \mathfrak{S}_\gamma\pi$ that is lost through a censoring process. Assuming uniformly random censoring in a parametric setting, we obtain the following observed likelihood with respect to the partially ranked data set D

$$\ell(\theta|D) = \sum_{i=1}^{m} \log \frac{1}{|\mathfrak{S}_{\gamma_i\pi_i}|} \sum_{\sigma \in \mathfrak{S}_{\gamma_i}\pi_i} p_\theta(\sigma) = \sum_{i=1}^{m} \log \sum_{\sigma \in \mathfrak{S}_{\gamma_i}\pi_i} p_\theta(\sigma) + \text{const.}$$

While the above likelihood function can be efficiently computed using tools developed in Section 7, its maximization is extremely difficult due to the discrete nature of the parametric space. In the next section we explore in detail a non-parametric kernel smoothing alternative to estimating p and g based on partially ranked data.

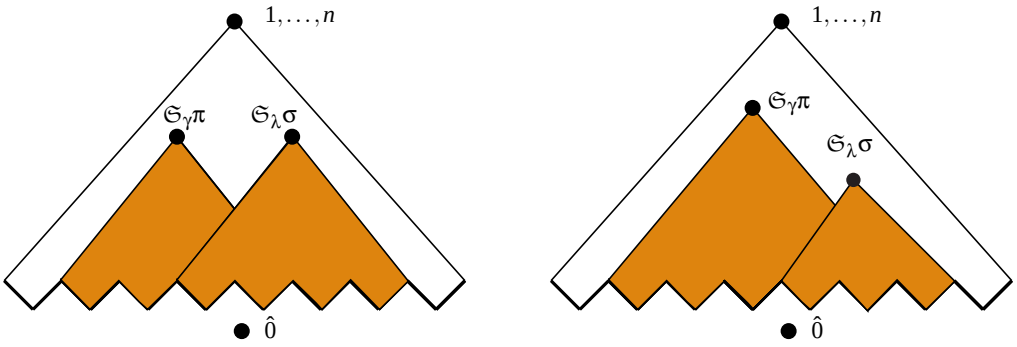


Figure 3: Two partial rankings with the same (left) and different (right) number of vertical bars in the Hasse diagram of $\tilde{\mathfrak{W}}_n$. The big triangles are schematic illustration of the Hasse diagram of $\mathfrak{W}_n$ as displayed in Figure 2 (top) with permutations occupying the bottom level illustrated by the jagged line. The shaded regions correspond to order-intervals, that is, all elements smaller or equal to the top vertices which correspond to partial rankings. To avoid probabilistic contradictions, the values of g at two non-disjoint partial rankings $\mathfrak{S}_\gamma\pi, \mathfrak{S}_\lambda\sigma$ cannot be specified in an independent manner.

6. Non-Parametric Kernel Smoothing on Partial Rankings

The Mallows model, which at first glance appears as a simple and effective analogue of the Gaussian distribution, suffers from several drawbacks. Its unimodal assumption is often too restrictive for high n as well as for low n (see experiments in Section 9). Another major drawback is that the location parameter space $\mathfrak{S}_n$ is discrete, making the maximum likelihood procedure an impossibly large discrete search problem.

The unimodality and symmetry of the Mallows model make it a good choice for use as a kernel in non-parametric smoothing. Since the normalization term ψ does not depend on the location parameter (6), the kernel smoothing estimator for p is

$$\hat{p}(\pi) = \frac{1}{m\,\psi(c)} \sum_{i=1}^{m} \exp(-c\,d(\pi,\pi_i)) \quad \pi,\pi_i \in \mathfrak{S}_n \tag{11}$$

assuming the data consists of complete rankings $\pi_1,\ldots,\pi_m \sim p$. Note that the kernel parameter c acts as an inverse scale parameter whose role is similar but inversely related to the traditional bandwidth parameter h in kernel smoothing (Wand and Jones, 1995).

In case the available data is partially ranked $D = \{\mathfrak{S}_{\gamma_i}\pi_i : i = 1,\ldots,m\}$ and obtained by uniform censoring as described in the previous section the kernel smoothing estimator becomes

$$\hat{p}(\pi) = \frac{1}{m\,\psi(c)} \sum_{i=1}^{m} \frac{1}{|\mathfrak{S}_{\gamma_i}|} \sum_{\tau\in\mathfrak{S}_{\gamma_i}\pi_i} \exp(-c\,d(\pi,\tau)) \qquad \pi \in \mathfrak{S}_n \tag{12}$$

where we used the fact that $|\mathfrak{S}_{\gamma_i}\pi_i| = |\mathfrak{S}_{\gamma_i}e| = |\mathfrak{S}_{\gamma_i}|$. The lattice or partial ranking version $\hat{g}$ corresponding to $\hat{p}$ in (12) is

$$\hat{g}(\mathfrak{S}_\lambda \pi) = \frac{1}{m\,\psi(c)} \sum_{i=1}^{m} \frac{1}{|\mathfrak{S}_{\gamma_i}|} \sum_{\kappa \in \mathfrak{S}_\lambda \pi} \sum_{\tau \in \mathfrak{S}_{\gamma_i}\pi_i} \exp(-c\,d(\kappa,\tau)) \qquad \mathfrak{S}_\gamma \pi \in \tilde{\mathfrak{W}}_n. \tag{13}$$

In the next section we derive efficient calculations and in some cases closed forms for expressions (12)-(13). These calculations are efficient even for large n as their complexities depend on the complexity of the compositions λ and $\gamma_1,\ldots,\gamma_m$ rather than on $n!$ or even n. We then move on to explore the bias and variance of $\hat{p}$ in Section 8 and describe practical applications of $\hat{p},\hat{g}$ and some experiments.

7. Efficient Computation and Inversion Combinatorics

In order to apply the estimators $\hat{p},\hat{g}$ in practice, it is crucial that the inner summations in Equations (12)-(13) be computed efficiently. We can achieve efficient computation of these summations by considering how the pairs constituting inversions $i(\tau)$ decompose with respect to certain cosets.

Proposition 4 *The following decomposition of $i(\tau)$ with respect to a composition $\gamma = (\gamma_1,\ldots,\gamma_r)$ holds*

$$i(\tau) = \sum_{k=1}^{r} a_k^\gamma(\tau) + \sum_{k=1}^{r}\sum_{l=k+1}^{r} b_{kl}^\gamma(\tau) \qquad \forall \tau \in \mathfrak{S}_n \tag{14}$$

where

$$a_k^\gamma(\tau) \stackrel{\text{def}}{=} \left| \left\{ (s,t) : s<t\,, \sum_{j=1}^{k-1}\gamma_j < \tau(t) < \tau(s) \le \sum_{j=1}^{k}\gamma_j \right\} \right|$$

$$b_{kl}^\gamma(\tau) \stackrel{\text{def}}{=} \left| \left\{ (s,t) : s<t\,, \sum_{j=1}^{k-1}\gamma_j < \tau(t) \le \sum_{j=1}^{k}\gamma_j \le \sum_{j=1}^{l-1}\gamma_j < \tau(s) \le \sum_{j=1}^{l}\gamma_j \right\} \right|.$$

Proof The set appearing in the definition of $a_k^\gamma(\tau)$ contains all pairs (s,t) that are inversions of τ and whose ranks appear in the k-compartment of the composition γ. The set appearing in the definition of $b_{kl}^\gamma(\tau)$ contains pairs (s,t) that are inversions of τ and for which s and t appear in the l and k compartments of γ respectively. Since any inversion pair appears in either one or two compartments, the above forms a partition of the inversion set. The decomposition holds since $i(\tau)$, the cardinality of the inversion set of the permutation τ, equals the summation of the cardinality of each subset in the partition. ■

Equation (14) actually represents a family of decompositions as it holds for all possible compositions γ. For example, $i(\tau) = 4$ for $\tau = 4|1|3|2$, with inversions $(1,4),(2,4),(3,4),(2,3)$ for τ. For the composition $\gamma = (2,2)$, the first compartment contains the inversion $(1,4)$ and so $a_1^\gamma(\tau) = 1$. The second compartment contains the inversion $(2,3)$ and so $a_2^\gamma(\tau) = 1$. The cross compartment inversions are $(2,4),(3,4)$ making $b_{12}^\gamma(\tau) = 2$.

The significance of (14) is that as we sum over all representatives of the coset $\tau \in \mathfrak{S}_\gamma\pi$ the cross compartmental inversions $b_{kl}^\gamma(\tau)$ remain constant while the within-compartmental inversions

$a_k^\gamma(\tau)$ vary over all possible combinations. As a result we obtain the following generalization of Proposition 2.

Proposition 5 *For $\pi \in \mathfrak{S}_n$, $q > 0$, and a composition $\gamma = (\gamma_1, \ldots, \gamma_r)$ we have*

$$\sum_{\tau \in \mathfrak{S}_\gamma \pi} q^{i(\tau)} = q^{\sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi)} \prod_{s=1}^r \prod_{j=1}^{\gamma_s - 1} \sum_{k=0}^j q^k. \tag{15}$$

Proof

$$\begin{aligned}
\sum_{\tau \in \mathfrak{S}_\gamma \pi} q^{i(\tau)} &= \sum_{\tau \in \mathfrak{S}_\gamma \pi} q^{\sum_{k=1}^r a_k^\gamma(\tau) + \sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\tau)} \\
&= q^{\sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi)} \sum_{\tau \in \mathfrak{S}_\gamma \pi} q^{\sum_{k=1}^r a_k^\gamma(\tau)} \\
&= q^{\sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi)} \prod_{s=1}^r \sum_{\tau \in \mathfrak{S}_{\gamma_s}} q^{i(\tau)} \\
&= q^{\sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi)} \prod_{s=1}^r \prod_{j=1}^{\gamma_s - 1} \sum_{k=0}^j q^k.
\end{aligned}$$

Above, we used two ideas: (i) disconcordant pairs between two different compartments of the coset $\mathfrak{S}_\gamma \pi$ are invariant under change of the coset representative, and (ii) the number of disconcordant pairs within a compartment varies over all possible choices enabling the replacement of the summation by a sum over a lower order symmetric group. ■

An important feature of (15) is that only the first and relatively simple term $q^{\sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi)}$ depends on π. The remaining terms depend only on the partial ranking type γ and thus may be pre-computed and tabulated for efficient computation.

Corollary 1

$$\sum_{\tau \in \mathfrak{S}_\gamma \pi} q^{i(\tau\kappa)} = q^{\sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi\kappa)} \prod_{s=1}^r \prod_{j=1}^{\gamma_s - 1} \sum_{k=0}^j q^k \qquad \kappa \in \mathfrak{S}_n.$$

Proof Using group theory, it can be shown that the set equality $(\mathfrak{S}_\gamma \pi)\kappa = \mathfrak{S}_\gamma(\pi\kappa)$ holds. As a result, $\sum_{\tau \in \mathfrak{S}_\gamma \pi} q^{i(\tau\kappa)} = \sum_{\tau' \in \mathfrak{S}_\gamma(\pi\kappa)} q^{i(\tau')}$. Proposition 5 completes the proof. ■

Corollary 2 *The partial ranking version g corresponding to the Mallows kernel p_κ is*

$$\begin{aligned}
p_\kappa(\mathfrak{S}_\gamma \pi) &= \frac{\prod_{s=1}^r \prod_{j=1}^{\gamma_s - 1} \sum_{k=0}^j e^{-kc}}{\prod_{j=1}^{n-1} \sum_{k=0}^j e^{-kc}} \, e^{-c \sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi\kappa^{-1})} \\
&\propto e^{-c \sum_{k=1}^r \sum_{l=k+1}^r b_{kl}^\gamma(\pi\kappa^{-1})}.
\end{aligned}$$

Proof Using Corollary 1 we have

$$\begin{aligned} g(\mathfrak{S}_\gamma\pi) &= \sum_{\tau\in\mathfrak{S}_\gamma\pi} p_\kappa(\tau) = \frac{\sum_{\tau\in\mathfrak{S}_\gamma\pi}\exp(-c\,d(\tau,\kappa))}{\sum_{\tau\in\mathfrak{S}_n}\exp(-c\,d(\tau,\kappa))} \\ &= \frac{\sum_{\tau\in\mathfrak{S}_\gamma\pi}\exp(-c\,i(\tau\kappa^{-1}))}{\prod_{j=1}^{n-1}\sum_{k=0}^{j}e^{-kc}} = \frac{\sum_{\tau\in\mathfrak{S}_\gamma\pi}(\exp(-c))^{i(\tau\kappa^{-1})}}{\prod_{j=1}^{n-1}\sum_{k=0}^{j}e^{-kc}} \\ &= e^{-c\sum_{k=1}^{r}\sum_{l=k+1}^{r}b_{kl}^{\gamma}(\pi\kappa^{-1})}\frac{\prod_{s=1}^{r}\prod_{j=1}^{\gamma_s-1}\sum_{k=0}^{j}e^{-kc}}{\prod_{j=1}^{n-1}\sum_{k=0}^{j}e^{-kc}}. \end{aligned}$$

■

Despite its daunting appearance, the expression in Corollary 2 can be computed relatively easily. The fraction does not depend on π or κ and in fact may be considered as a normalization constant that may be easily pre-computed and tabulated. The remaining term is relatively simple and depends on the location parameter κ and the coset representative π. Corollary 2 and Proposition 6 below, provide efficient computation for the estimators (12), (13).

Proposition 6

$$\sum_{\sigma\in\mathfrak{S}_\lambda\pi_1}\sum_{\tau\in\mathfrak{S}_\gamma\pi_2}e^{-c\,d(\sigma,\tau)} = \left(\sum_{\tau\in\pi_1\pi_2^{-1}\mathfrak{S}_\gamma}\prod_{k=1}^{r}\prod_{l=k+1}^{r}e^{-c\,b_{kl}^{\lambda}(\tau)}\right)\left(\prod_{s=1}^{r}\prod_{j=1}^{\lambda_s-1}\sum_{k=0}^{j}e^{-kc}\right). \tag{16}$$

Proof Using $(\mathfrak{S}_\gamma\pi)\tau = \mathfrak{S}_\gamma(\pi\tau)$, Corollary 1, and the fact that $\tau\in\mathfrak{S}_\gamma$ iff $\tau^{-1}\in\mathfrak{S}_\gamma$, we have

$$\begin{aligned} \sum_{\sigma\in\mathfrak{S}_\lambda\pi_1}\sum_{\tau\in\mathfrak{S}_\gamma\pi_2}e^{-cd(\sigma,\tau)} &= \sum_{\sigma\in\mathfrak{S}_\lambda}\sum_{\tau\in\mathfrak{S}_\gamma}e^{-cd(\sigma\pi_1,\tau\pi_2)} = \sum_{\sigma\in\mathfrak{S}_\lambda}\sum_{\tau\in\mathfrak{S}_\gamma}e^{-cd(\sigma\pi_1\pi_2^{-1}\tau^{-1},e)} \\ &= \sum_{\tau\in\mathfrak{S}_\gamma}\sum_{\sigma\in\mathfrak{S}_\lambda}e^{-c\,i(\sigma\pi_1\pi_2^{-1}\tau^{-1})} = \sum_{\tau\in\mathfrak{S}_\gamma}\sum_{\sigma\in\mathfrak{S}_\lambda}e^{-c\,i(\sigma(\pi_1\pi_2^{-1})\tau)} \\ &= \sum_{\tau\in\mathfrak{S}_\gamma}e^{-c\sum_{k=1}^{r}\sum_{l=k+1}^{r}b_{kl}^{\lambda}(\pi_1\pi_2^{-1}\tau)}\prod_{s=1}^{r}\prod_{j=1}^{\lambda_s-1}\sum_{k=0}^{j}e^{-kc}. \end{aligned}$$

■

The complexity of computing (16), (12), (13) for some popular partial ranking types appears in Table 1. The independence of these complexity terms from n enables the practical use of estimators (12), (13) in large n situations. Some of the details concerning this complexity analysis and algorithmic implementation may be found in Appendix A.

8. Statistical Properties of the Estimator

After studying the computational feasibility of the non-parametric estimator $\hat{p}$ in the previous section, we now turn to examine its statistical properties. In particular we examine its bias and variance,

$\lambda \diagdown \gamma$	$(1,n-1)$	$(1,\cdots,1,n-t)$	$(t,n-t)$
$(1,n-1)$	$O(1)$	$O(1)$	$O(1)$
$(1,\cdots,1,n-k)$	$O(k)$	$O(k+t)$	$O(k+t)$
$(k,n-k)$	$O(k)$	$O(k+t)$	$O(k+t)$

Table 1: Computational complexity for computing Equation (13) for each training example. The independence of the complexity terms from n enables the practical use of the estimators (12),(13) in $k \ll n$ situations.

show consistency for large n, and examine the statistical effect of using partially ranked or censored data in the estimation process. Due to the discreteness of the probability space we replace traditional Taylor series expansion with a bound based on the Lipschitz continuity of p. The Lipschitz continuity assumption is crucial since without such an assumption on the regularity of p, kernel based smoothing or other neighborhood operations make little sense.

Proposition 7 *Let $\pi_1,\ldots,\pi_m \in \mathfrak{S}_n$ be sampled iid from a Lipschitz continuous p, that is, $|p(\pi)-p(\tau)| \leq M\,d(\pi,\tau)$, $\forall \pi,\tau$. The following bounds with respect to $\hat{p}$ in* (11) *hold.*

$$|\mathit{bias}\,(\hat{p}(\pi))| \leq -M\frac{\psi'(c)}{\psi(c)},$$
$$\mathit{Var}\,(\hat{p}(\pi)) \leq \frac{p(\pi)}{m}\frac{\psi(2c)}{\psi^2(c)} - \frac{M}{m}\frac{\psi'(2c)}{\psi^2(c)}.$$

Proof Key properties in the following manipulations are the closed form expression of $\psi(c)$ in (6) and its independence from the location parameter of the Mallows kernel.

$$\begin{aligned}
|\mathsf{bias}\,(\hat{p}(\pi))| &= \left|\mathsf{E}_{p(\pi_1)}\left(\psi^{-1}(c)e^{-cd(\pi,\pi_1)}\right) - p(\pi)\right| \\
&\leq \psi^{-1}(c)\sum_{\pi_1\in\mathfrak{S}_n}|p(\pi_1)-p(\pi)|e^{-cd(\pi,\pi_1)} \\
&\leq \psi^{-1}(c)\sum_{\pi_1\in\mathfrak{S}_n}Md(\pi,\pi_1)e^{-cd(\pi,\pi_1)} = -M\frac{\psi'(c)}{\psi(c)}.
\end{aligned}$$

$$\begin{aligned}
\psi^2(c)\,m\,\mathsf{Var}\,(\hat{p}(\pi)) &= \mathsf{Var}_{\,p(\pi_1)}e^{-c\,d(\pi,\pi_1)} \leq \mathsf{E}_{\,p(\pi_1)}e^{-2cd(\pi,\pi_1)} \\
&= \sum_{\pi_1\in\mathfrak{S}_n}p(\pi_1)e^{-2c\,d(\pi,\pi_1)} \\
&\leq \sum_{\pi_1\in\mathfrak{S}_n}(p(\pi)+M\,d(\pi,\pi_1))e^{-2cd(\pi,\pi_1)} \\
&= p(\pi)\psi(2c) - M\psi'(2c).
\end{aligned}$$

■

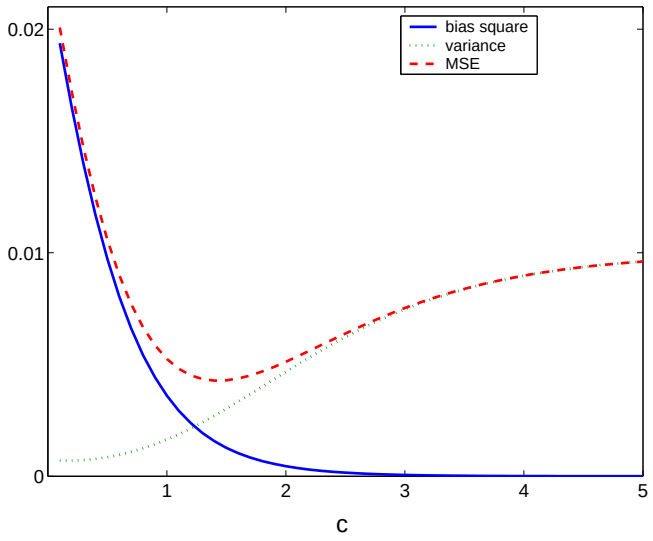


Figure 4: Upper bounds on squared bias, variance and MSE as functions of c: $M = 0.05$, $p(\pi) = 0.2$, $n = 4$, $m = 20$.

The upper bounds in Proposition 7 are illustrated as functions of c in Figure 4. These expressions may be written in a closed form using the formulas for $\psi(2c)/\psi^2(c)$ and $\psi'(c)/\psi(c)$ derived in the proof of Proposition 8 below.

Proposition 8 *Under the same conditions as Proposition 7 and assuming the asymptotics*

$$c, m, n \to \infty, \quad n = o(\exp(c)), \quad n = o(\sqrt{m})$$

the estimator $\hat{p}$ in (11) *is pointwise consistent.*

Proof We first derive closed form expressions for $\psi'(c)/\psi(c)$ and $\psi(2c)/\psi^2(c)$ and then proceed to demonstrate the convergence to 0 of the bias and variance bounds obtained in Proposition 7.

Using the result $\psi(c) = \prod_{j=1}^{n} \frac{1-e^{-jc}}{1-e^{-c}}$ shown in (6), we have

$$\frac{\psi'(c)}{\psi(c)} = (\log \psi(c))' = \sum_{j=1}^{n} \frac{j e^{-jc}}{1-e^{-jc}} - \frac{n e^{-c}}{1-e^{-c}}, \tag{17}$$

$$\begin{aligned} \frac{\psi(2c)}{\psi^2(c)} &= \prod_{j=1}^{n} \frac{1-e^{-2jc}}{1-e^{-2c}} \frac{(1-e^{-c})^2}{(1-e^{-jc})^2} = \prod_{j=1}^{n} \frac{1+e^{-jc}}{1+e^{-c}} \frac{1-e^{-jc}}{1-e^{-c}} \frac{(1-e^{-c})^2}{(1-e^{-jc})^2} \\ &= \prod_{j=1}^{n} \frac{1+e^{-jc}}{1-e^{-jc}} \frac{1-e^{-c}}{1+e^{-c}}. \end{aligned} \tag{18}$$

The term $-\psi'(c)/\psi(c)$ is the expected distance under the Mallows model

$$-\psi'(c)/\psi(c) = \sum_{\sigma \in \mathfrak{S}_n} d(\pi, \sigma) \psi^{-1}(c) \exp(-c\, d(\pi, \sigma))$$

and therefore is bounded by $\max_{\pi,\sigma} d(\pi,\sigma) \leq n^2$. The term $\psi(2c)/\psi^2(c)$ is bounded since it may be written as a product $\prod_{j=1}^{n} R_j(c)$, with $R_j(c) = \frac{1+e^{-jc}}{1-e^{-jc}} / \frac{1+e^{-c}}{1-e^{-c}} \leq 1$ for all $c \in \mathbb{R}_+$ and $j \geq 1$ since the function $\frac{1+\varepsilon}{1-\varepsilon}$ increases with $\varepsilon > 0$.

Based on Proposition 7 and Equations (17)-(18)

$$|\mathrm{bias}\,(\hat{p}(\pi))| \leq M\frac{ne^{-c}}{1-e^{-c}} - M\sum_{j=1}^{n}\frac{je^{-jc}}{1-e^{-jc}} \leq M\frac{ne^{-c}}{1-e^{-c}}$$

$$\mathrm{Var}\,(\hat{p}(\pi)) \leq \frac{p(\pi)}{m}\frac{\psi(2c)}{\psi^2(c)} - \frac{M}{m}\frac{\psi'(2c)}{\psi(2c)}\frac{\psi(2c)}{\psi^2(c)}.$$

The bias converges to 0 as $n\exp(-c) \to 0$ or alternatively, $c \to \infty, n = o(\exp(c))$. Since $\psi(2c)/\psi^2(c)$ is bounded and $-\psi'(2c)/\psi(2c) \leq n^2$ the variance converges to 0 as well if $m \to \infty$ and $n^2/m \to 0$. ■

Intuitively, the inverse scale parameter c has to go to ∞ in order for the bias to converge to 0 (similar to the requirement $h \to 0$ for the bandwidth parameter h in kernel smoothing). The number of samples m has to go to ∞ in order for the variance to go to 0. Allowing $n \to \infty$ enables us to study the behavior of $\hat{p}$ in situations containing a large number of items. The proposition above (with a slightly modified proof) also holds for fixed n.

The assumption above of Lipschitz continuity with respect to d is a very weak assumption since the distance d tends to grow as $n \to \infty$. In particular d takes values in $[0, n(n-1)/2]$ making the Lipschitz continuity assumption weaker and weaker as $n \to \infty$. A stronger assumption of Lipschitz continuity with respect to the normalized d_n (5)

$$|p(\pi) - p(\tau)| \leq M\, d_n(\pi,\tau), \quad \forall \pi,\tau$$

results in a similar conclusion to Proposition 8 asserting pointwise consistency of $\hat{p}$ under weaker asymptotic requirements.

For large n, it is often the case that partial, rather than full, rankings are available for estimating $\hat{p}$. Partially ranked data is easier for rankers to express than a lengthy list corresponding to a precise permutation. Furthermore, in many cases, rankers can make some partial ranking assertions with certainty but do not have a clear opinion on other preferences. Using the censored data interpretation of partially ranked data enables efficient use of partially ranked data of multiple types in the estimation process (12).

Statistically, expressing partially ranked data as censored data has the effect of increased smoothing and therefore it reduces the variance while increasing the bias. The following proposition quantifies this effect in terms of the bias and variance of $\hat{p}$. A consequence of this proposition which is also illustrated in Section 9 experimentally is that even if the fully ranked data is somehow available, estimating $\hat{p}$ based on the partial rankings obtained by censoring it tends to increase the estimation accuracy.

Proposition 9 *Assuming the same conditions as in Proposition 7, the bias and variance of the censored data or partial ranking estimator* (12) *for* $\gamma_1 = \ldots = \gamma_m = \gamma$ *satisfy*

$$|bias\,(\hat{p}(\pi))| \leq -M\frac{\psi'(c)}{\psi(c)} + M\frac{sp(\mathfrak{S}_\gamma)}{|\mathfrak{S}_\gamma|},$$
$$Var\,(\hat{p}(\pi)) \leq \frac{p(\pi)}{m}\frac{1}{|\mathfrak{S}_\gamma|} + \frac{M}{m}\frac{sp(\mathfrak{S}_n)}{|\mathfrak{S}_\gamma|^2} \tag{19}$$

where $sp(U) \stackrel{\text{def}}{=} \max_{x\in U}\sum_{y\in U} d(x,y)$.

The choice of using $\gamma_1 = \ldots = \gamma_m = \gamma$ above was made for simplicity. Similar results apply for more heterogenous partially ranked data.

Proof

$$\begin{aligned}
|\mathrm{bias}\,(\hat{p}(\pi))| &= \left|\psi^{-1}(c)|\mathfrak{S}_\gamma|^{-1}\mathsf{E}_{p(\pi_1)}\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} e^{-cd(\pi,\tau)} - p(\pi)\right| \\
&\stackrel{a}{\leq} \psi^{-1}(c)|\mathfrak{S}_\gamma|^{-1}\sum_{\pi_1\in\mathfrak{S}_n}\sum_{\tau\in\mathfrak{S}_\gamma\pi_1}|p(\pi_1)-p(\pi)|e^{-cd(\pi,\tau)} \\
&\leq M\psi^{-1}(c)|\mathfrak{S}_\gamma|^{-1}\sum_{\pi_1\in\mathfrak{S}_n}\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} d(\pi,\pi_1)e^{-cd(\pi,\tau)} \\
&\leq M\psi^{-1}(c)|\mathfrak{S}_\gamma|^{-1}\sum_{\pi_1\in\mathfrak{S}_n}\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} (d(\pi,\tau)+d(\tau,\pi_1))e^{-cd(\pi,\tau)} \\
&\stackrel{b}{=} -M\frac{\psi'(c)}{\psi(c)} + \frac{M}{\psi(c)|\mathfrak{S}_\gamma|}\sum_{\pi_1\in\mathfrak{S}_n}\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} d(\tau,\pi_1)e^{-cd(\pi,\tau)}
\end{aligned}$$

where *a* and *b* follow from the fact that

$$\sum_{\pi_1\in\mathfrak{S}_n}\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} e^{-cd(\pi,\tau)} = |\mathfrak{S}_\gamma|\sum_{\tau\in\mathfrak{S}_n} e^{-cd(\pi,\tau)} = |\mathfrak{S}_\gamma|\psi(c).$$

The inner summation depends on π_1 only through the coset $\mathfrak{S}_\gamma\pi_1$ it resides in. To simplify the expression, we separate the single outer summation to summations of π_1 over the distinct cosets C_j. Since the number of distinct $\mathfrak{S}_\gamma$ cosets in $\mathfrak{S}_n$ is the index $[\mathfrak{S}_n : \mathfrak{S}_\gamma] = |\mathfrak{S}_n|/|\mathfrak{S}_\gamma|$, we have

$$\begin{aligned}
|\mathrm{bias}\,(\hat{p}(\pi))| &\leq -M\frac{\psi'(c)}{\psi(c)} + \frac{M}{\psi(c)|\mathfrak{S}_\gamma|}\sum_{j=1}^{[\mathfrak{S}_n:\mathfrak{S}_\gamma]}\sum_{\tau\in C_j}\left(\sum_{\pi_1\in C_j} d(\tau,\pi_1)\right)e^{-cd(\pi,\tau)} \\
&\leq -M\frac{\psi'(c)}{\psi(c)} + \frac{M\,\mathrm{sp}(\mathfrak{S}_\gamma)}{\psi(c)|\mathfrak{S}_\gamma|}\sum_{j=1}^{[\mathfrak{S}_n:\mathfrak{S}_\gamma]}\sum_{\tau\in C_j} e^{-cd(\pi,\tau)} \\
&= -M\frac{\psi'(c)}{\psi(c)} + M\frac{\mathrm{sp}(\mathfrak{S}_\gamma)}{|\mathfrak{S}_\gamma|}
\end{aligned}$$

using the fact that the spread is the same for all cosets of the same type $\mathrm{sp}(\mathfrak{S}_\gamma\pi) = \mathrm{sp}(\mathfrak{S}_\gamma)$.

$$
\begin{aligned}
m\psi^2(c)\,|\mathfrak{S}_\gamma|^2\,\mathsf{Var}\,(\hat{p}(\pi)) &= \mathsf{Var}_{\,p(\pi_1)} \sum_{\tau\in\mathfrak{S}_\gamma\pi_1} e^{-cd(\pi,\tau)} \le \mathsf{E}_{\,p(\pi_1)} \left(\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} e^{-cd(\pi,\tau)} \right)^2 \\
&\le \sum_{\pi_1\in\mathfrak{S}_n} (p(\pi) + Md(\pi,\pi_1)) \left(\sum_{\tau\in\mathfrak{S}_\gamma\pi_1} e^{-cd(\pi,\tau)} \right)^2 \\
&= \sum_{j=1}^{[\mathfrak{S}_n:\mathfrak{S}_\gamma]} \left(\sum_{\tau\in C_j} e^{-cd(\pi,\tau)} \right)^2 \left(p(\pi)|\mathfrak{S}_\gamma| + M \sum_{\sigma\in C_j} d(\pi,\sigma) \right) \\
&\le p(\pi)|\mathfrak{S}_\gamma|\psi^2(c) + M\mathrm{sp}(\mathfrak{S}_n)\psi^2(c).
\end{aligned}
$$

In the last inequality we used the Cauchy-Schwartz inequality $\langle u,v\rangle \le \|u\|_2\|v\|_2 \le \|u\|_1\|v\|_1$ to obtain

$$
\sum_{j=1}^{[\mathfrak{S}_n:\mathfrak{S}_\gamma]} \left(\sum_{\tau\in C_j} e^{-cd(\pi,\tau)} \right)^2 \le \left(\sum_{j=1}^{[\mathfrak{S}_n:\mathfrak{S}_\gamma]} \sum_{\tau\in C_j} e^{-cd(\pi,\tau)} \right)^2 = \psi^2(c).
$$

■

Contrasting the expressions in Proposition 9 with those in Proposition 7 indicates that reverting to partial rankings tends to increase the bias but reduce the variance. Intuitively, the bias increases since we no longer have enough data, in general, to precisely estimate the permutation model p. The variance (19), on the other hand, experiences a substantial reduction as compared to the fully ranked case. Figure 5 displays the behavior of the quantities $\frac{\mathrm{sp}(\mathfrak{S}_\gamma)}{|\mathfrak{S}_\gamma|}$ and $\frac{\mathrm{sp}(\mathfrak{S}_n)}{|\mathfrak{S}_\gamma|^2}$. The first quantity $\frac{\mathrm{sp}(\mathfrak{S}_\gamma)}{|\mathfrak{S}_\gamma|}$, which bounds the bias, increases as the composition γ represents a lower degree of specificity. On the other hand, the second quantity $\frac{\mathrm{sp}(\mathfrak{S}_n)}{|\mathfrak{S}_\gamma|^2}$ which bounds the variance decreases as the composition γ represents less specificity.

The precise changes in the bias and variance that occur due to using partial rankings depend on γ, n, m, c, M. However, generally speaking, the variance reduction becomes more pronounced as n and $|\mathfrak{S}_\gamma|$ grow. Indeed, in the common case described earlier where the number of items n is large, switching to partially ranked data can dramatically improve the estimation accuracy. This observation, which is illustrated in Section 9 using experiments on real world data, becomes increasingly important as n increases. It is remarkable that this statistical motivation to use partial rather than full rankings is aligned with the data availability and ease of use as well as with the computational efficiency demonstrated in the previous section.

9. Applications and Experiments

The estimator $\hat{p}$ defined in (11), (12) and its lattice version $\hat{g}$ defined in (13) can be used in a number of data analysis tasks. We briefly outline some of these tasks below and then proceed to describe some experimental results.

Visual or computational exploration of the model probabilities $\{\hat{p}(\pi) : \pi \in \mathfrak{S}_n\}$ can be a useful exploratory data analysis tool. Such exploration can be done by visualizing the values $\{\hat{p}(\pi) : \pi \in \mathfrak{S}_n\}$ for small n using the techniques developed in Thompson (1993). For medium and large n

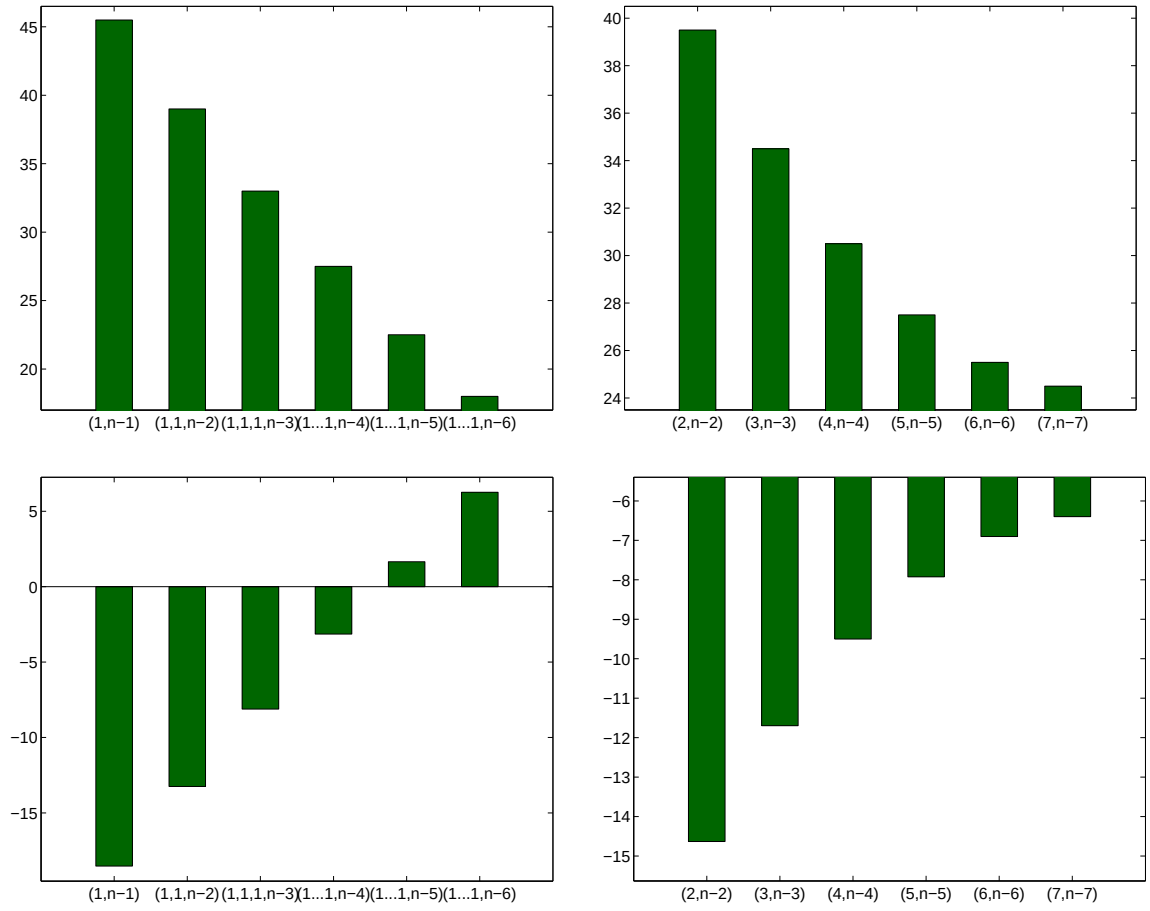


Figure 5: Values of $\frac{\mathrm{sp}(\mathfrak{S}_\gamma)}{|\mathfrak{S}_\gamma|}$(top row), and $\log \frac{\mathrm{sp}(\mathfrak{S}_n)}{|\mathfrak{S}_\gamma|^2}$ (bottom row) for $n = 15$ and various partial ranking types. Note that $\frac{\mathrm{sp}(\mathfrak{S}_\gamma)}{|\mathfrak{S}_\gamma|}$ (which serves as a bound for the bias) decreases and $\frac{\mathrm{sp}(\mathfrak{S}_n)}{|\mathfrak{S}_\gamma|^2}$ (which serves as a bound for the variance) increases for decreasing $|\mathfrak{S}_\gamma|$.

similar visualization techniques can be used to explore the values of the lattice version $\hat{g}$ restricted to certain subset $H \subset \tilde{\mathfrak{W}}_n$ of the ranking lattice. Since the number of distinct γ-cosets $|\mathfrak{S}_n|/|\mathfrak{S}_\gamma|$ may be much smaller than $|\mathfrak{S}_n|$, visualizing $\{\hat{g}(A) : A \in H\}$ can be more effective than visualizing $\{\hat{p}(\pi) : \pi \in \mathfrak{S}_n\}$. Other explorations such as identifying the local modes of $\hat{p}$ and $\hat{g}$ may be automated and computed without human intervention.

In some cases, the main objective of inference is a conditional version of $\hat{p}$ such as $\hat{p}(\pi \in A | \pi \in B)$, $A, B \subset \mathfrak{S}_n$. A popular example is collaborative filtering which is the task of recommending items to a user based on partial preference information that is output by that user (Resnick et al., 1994). In this case, $\hat{p}$ is estimated based on a large data set of partial preferences provided by many users. Given a particular partial ranking $\mathfrak{S}_\gamma\pi$ output by a certain user we can predict its most likely refinement $\arg\max_{\mathfrak{S}_\lambda\sigma} \hat{p}(\mathfrak{S}_\lambda\sigma | \mathfrak{S}_\gamma\pi)$. This task is central to many recommendation systems and

has recently gained popularity in the machine learning research community due to its commercial applications.

Statistics such as expectations and variances can be useful as summaries in situations where the entire distribution is not necessary. For example, summaries such as the expectation and variance of an item's rank $\mathsf{E}_{\hat{p}}(\pi(k))$, $\mathsf{Var}_{\hat{p}}(\pi(k))$ or probabilities such as $\hat{p}(\pi(i) > \pi(j))$ may be useful in some cases. On the other hand, in situations where $\hat{p}$ is a complex multimodal distribution, the summaries need to be complemented with a careful examination of $\hat{p}$.

We experimented with three different data sets. The first is the APA data set (Diaconis, 1989) which contains several thousand rankings of 5 APA presidential candidates. The second is the Jester data set containing rankings of 100 jokes by 73,496 users. The third data set is the EachMovie data set containing rankings of 1628 movies by 72,916 users. In our experiments, we trained models based on a randomly sampled training set and evaluated the log-likelihood on a separate held-out testing set. We repeated this procedure 10 times and report the average log-likelihood in order to reduce sampling noise.

Figure 6 displays the test set log-likelihood for the parametric Mallows model (fitted by maximum likelihood) and the non-parametric estimator. The log-likelihood, computed as a function of the train set size, is displayed for several values of c for the non-parametric estimator. In the case of the Mallows model we only display the optimal c. Due to the computational difficulty associated with maximum likelihood for the Mallows model for large n we experimented with rankings over a small number of items. The three panels of the figure display the log-likelihood with respect to the APA data with $n = 5$ (top), the Jester data restricted to the $n = 5$ most frequently rated jokes (middle), and the EachMovie data restricted to the $n = 4$ most frequently rated movies. In all three cases, the non-parametric estimator performed better than the parametric Mallows model given sufficient training examples. As c increases, the non-parametric model tends to perform better for large data sets and worse for small data sets, reflecting the non-parametric consistency as $m, c \to \infty$.

The increased flexibility of the non-parametric model illustrated in Figure 6 can be visualized further by comparing the probabilities assigned by the Mallows model and the non-parametric model. We display these probabilities in the case of $n = 4$ (movies no. 357, 1356, 440, 25 from the EachMovie data) by scaling appropriately the vertices of the permutation polytope. The vertices of the permutation polytope, displayed in Figure 7, correspond to $\mathfrak{S}_4$ and its edges correspond to pairs of permutations with Kendall's tau distance 1. In fact, Kendall's tau distance $d(\pi, \sigma)$ corresponds to the length of the shortest path connecting the two vertices representing π and σ. As a result, the 3D embedding of the permutation polytope effectively visualizes the discrete metric space $(\mathfrak{S}_4, d)$. In the figure, the radiuses of the vertices were scaled proportionally to $(\hat{p}(\pi))^{5/7}$ where $\hat{p}(\pi)$ are the probabilities estimated by maximum likelihood Mallows model (left) and the non-parametric model (right). The scaling exponent of $5/7$ was chosen in agreement with Steven's law (Cleveland, 1985) for effective visualization. Figure 7 shows that the probabilities assigned by the Mallows model form a diffuse unimodal function centered at $2|1|3|4$. The non-parametric estimator, on the other hand, discovers the true global mode $2|3|1|4$ and an additional local mode at $4|1|2|3$ both of which were undiscovered by the Mallows model due to its unimodality property.

Figure 8 demonstrates non-parametric modeling of partial rankings for $n = 100$ (the Mallows model maximum likelihood estimator cannot be computed for such n). We used 10043 rankings from the Jester data set which contain users ranking all $n = 100$ jokes. As before, the figures display the test-set log-likelihood as a function of the train set size. Due to the large n, we measured the test

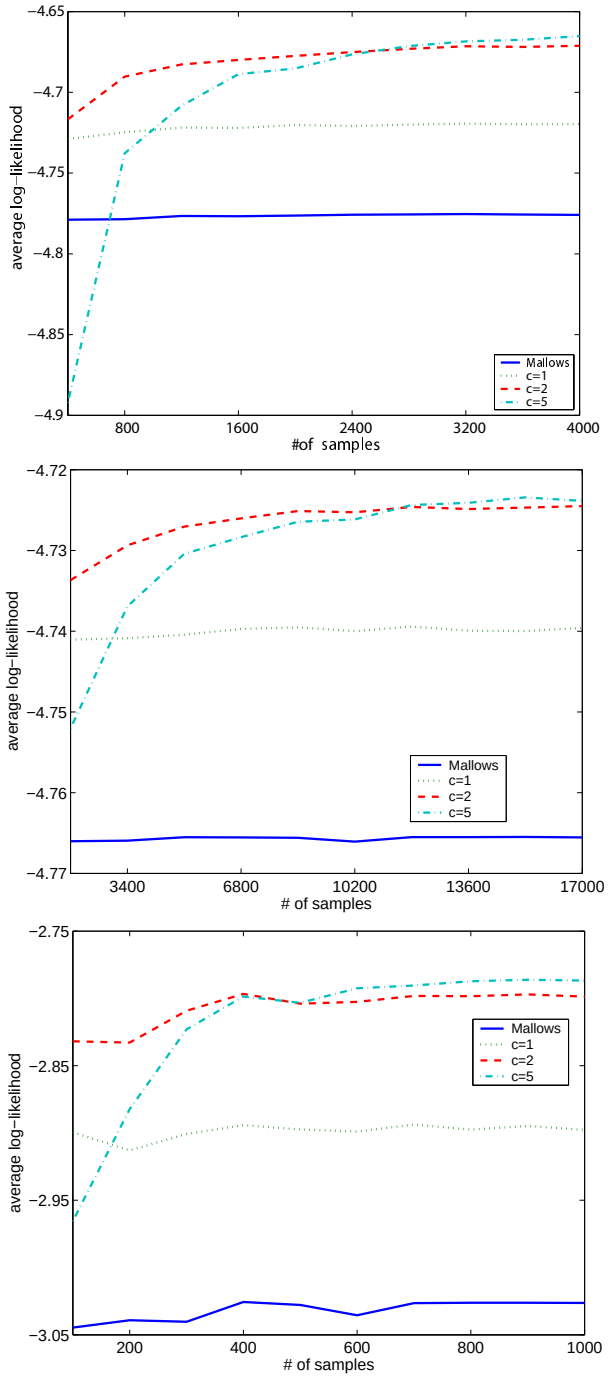


Figure 6: Average test log-likelihood as a function of the train set size: the maximum likelihood Mallows model vs. the non-parametric estimator for (a) APA data $n = 5$, (b) $n = 5$ most frequently rated Jester jokes, (c) $n = 4$ most frequently rated movies from EachMovie data. In general, the non-parametric model provides a better fit than the Mallows model. The non-parametric consistency is illustrated in the case of $c, m \to \infty$.

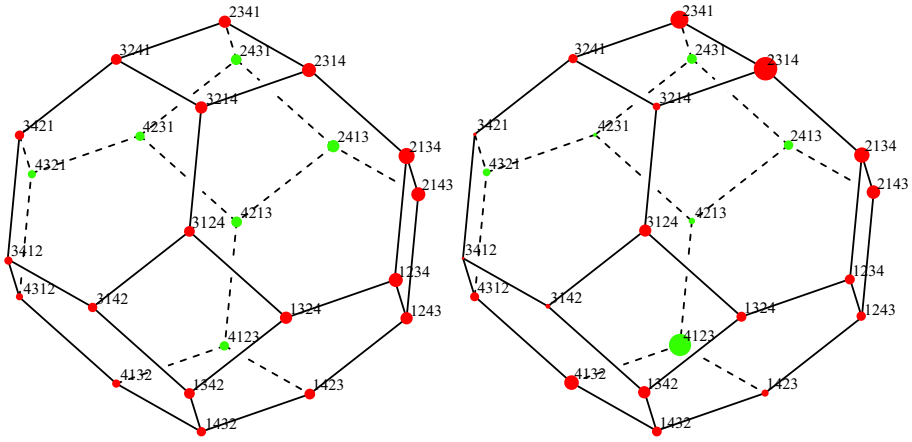


Figure 7: Visualizing estimated probabilities for EachMovie data by permutation polytopes: Mallows model (left) and non-parametric model for $c = 2$ (right). The Mallows model locates a single mode at $2|1|3|4$ while the non-parametric estimator locates the global mode at $2|3|1|4$ and a second local mode at $4|1|2|3$.

set log-likelihood with respect to the lattice version $\hat{g}(\mathfrak{S}_\gamma \pi)$ of the non-parametric estimator $\hat{p}$ for partial ranking $\gamma = (5, n-5)$ (top) and $\gamma = (1,1,1,n-3)$ (bottom).

The different lines in Figure 8 correspond to the performance of $\hat{p}$ obtained by censoring the training data in different ways. We compared $\hat{p}$ for the following censored data: full ranking (no censoring), $\gamma = (1,\ldots,1,n-k)$ for $k = 1,2,3,5$ and $\gamma = (k, n-k)$. The value of k in the censoring corresponding to $\gamma = (k, n-k)$ was chosen based on thresholding the scores output by the users. In particular, $(k(s), n-k(s))$ corresponds to k being the number of jokes receiving a score of s or higher (the users provided scores in the range $[-10, 10]$). The figure illustrates the statistical benefit of estimating $\hat{p}$ based on partial rather than full rankings. The variance reduction by $(k, n-k)$ partial rankings clearly outweighs the bias increase.

10. Discussion

As the number of items n increases, the space $\mathfrak{S}_n$ grows exponentially making discrete search methods such as maximum likelihood for the Mallows model difficult to compute. Similarly, it is typically the case for large n that both the data available for estimating $\hat{p}$ and the use of $\hat{p}$ will be restricted to partial rankings or cosets of the symmetric group.

Attempts to define a probabilistic model directly on multiple types of partial rankings $H \subset \tilde{\mathfrak{W}}_n$ face a challenging problem of preventing probabilistic contradictions. A simple solution is to define the partial ranking model $\hat{g}$ in terms of a permutation model $\hat{p}$ through the mechanism of Möbius inversion and censored data interpretation. However, doing so raises computational concerns that often severely limit the practical use of such models for large n.

In this paper, we present a non-parametric kernel smoothing technique that uses the Mallows model as a smoothing kernel on permutations. Using combinatorial properties of inversions and of the symmetric group we simplify the computational difficulties and exhibit its practical use inde-

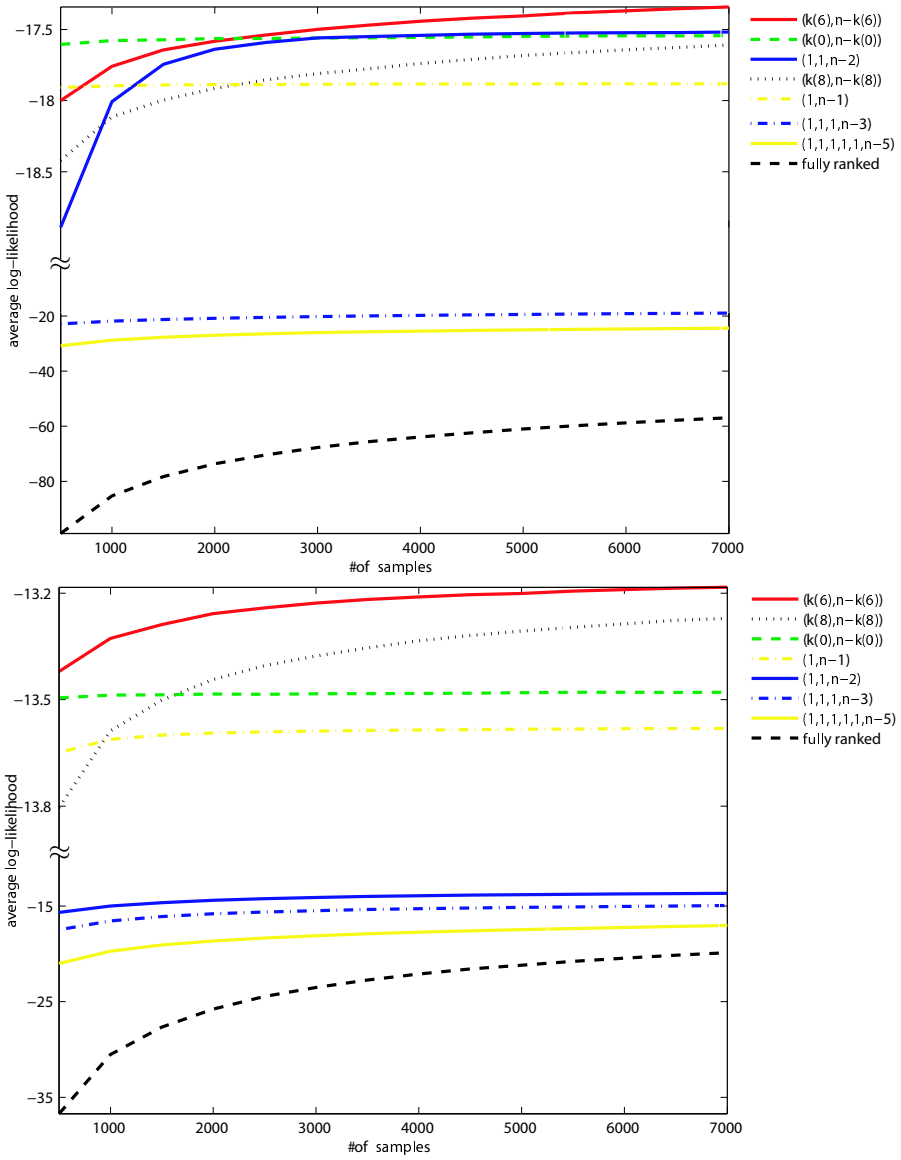


Figure 8: Test-set loglikelihood for $\hat{g}(\mathfrak{S}_\gamma\pi)$ with $\gamma=(5,n-5)$ (top) and $\gamma=(1,1,1,n-3)$ (bottom) as a function of train set size (Jester data set with $n=100$). The different lines correspond to obtaining $\hat{p}$ based on different censoring strategies of the fully ranked training data (see description in text). The legend entries are sorted in roughly the same order as the lines in the figures for increased visibility.

pendently of the number of items n. Theoretical and experimental examinations demonstrate the role of the inverse scale parameter c in the bias-variance tradeoff. We also examine the effect of using partial, rather than full, rankings on the bias and variance of the estimator. This effect plays a similar role to increased kernel smoothing and often leads to increased estimation accuracy.

Appendix A. Complexity Issues

Table 1 lists the computational complexity results for computing (13) for some popular partial ranking types γ and λ. The arguments or proofs for these expressions are rather involved and contain some details. We include in this appendix the details corresponding to the case of $\lambda = (k, n-k)$ and $\gamma = (t, n-t)$. The other cases in Table 1 follow similarly, but with some differences.

Proposition 10 *The complexity for computing*

$$\psi^{-1}(c)\,|\mathfrak{S}_\gamma|^{-1} \sum_{\sigma \in \mathfrak{S}_\lambda \pi_1} \sum_{\tau \in \mathfrak{S}_\gamma \pi_2} e^{-cd(\sigma,\tau)}$$

for $\lambda = (k, n-k)$ and $\gamma = (t, n-t)$ is $O(k+t)$.

Proof We first generalize the definition of cross compartment inversions $b^{\gamma}_{kl}(\tau)$ in Proposition 4 by defining

$$b_{XY}(\tau) = |\{(u,v) : u < v, \tau(v) \in X, \tau(u) \in Y\}|$$

where X and Y are arbitrary disjoint sets. If $X = \left\{ \sum_{j=1}^{k-1} \gamma_j + 1, \ldots, \sum_{j=1}^{k} \gamma_j \right\}$ and $Y = \left\{ \sum_{j=1}^{l-1} \gamma_j + 1, \ldots, \sum_{j=1}^{l} \gamma_j \right\}$, we have $b_{XY}(\tau) = b^{\gamma}_{kl}(\tau)$. If $x < y, \forall x \in X$ and $y \in Y$, $b_{XY}(\tau)$ counts a subset of inversion pairs of τ. However, in its most general form, $b_{XY}(\tau)$ may include non-inversion pairs if some numbers in X are greater than some numbers in Y.

We use the following definitions in our proof

$$\begin{aligned}
A &= \{1, \cdots, k\} \cap \{\pi_1 \pi_2^{-1}(1), \cdots, \pi_1 \pi_2^{-1}(t)\}, \\
\bar{A} &= \{1, \cdots, k\} \setminus A, \\
B &= \{k+1, \cdots, n\} \cap \{\pi_1 \pi_2^{-1}(1), \cdots, \pi_1 \pi_2^{-1}(t)\}, \\
\bar{B} &= \{k+1, \cdots, n\} \setminus B.
\end{aligned}$$

Note $A, \bar{A}, B, \bar{B}$ constitute a partition of $\{1, \ldots, n\}$ and satisfy

$$\begin{aligned}
A \cup \bar{A} &= \{1, \ldots, k\}, \\
B \cup \bar{B} &= \{k+1, \ldots, n\}, \\
A \cup B &= \{\pi_1 \pi_2^{-1}(1), \cdots, \pi_1 \pi_2^{-1}(t)\}, \\
\bar{A} \cup \bar{B} &= \{1, \cdots, n\} \setminus \{\pi_1 \pi_2^{-1}(1), \cdots, \pi_1 \pi_2^{-1}(t)\}.
\end{aligned}$$

Since $\lambda = (k, n-k)$, we have $b^{\lambda}_{12}(\tau) = b_{AB}(\tau) + b_{A\bar{B}}(\tau) + b_{\bar{A}B}(\tau) + b_{\bar{A}\bar{B}}(\tau)$, and the expression in the first parenthesis of Equation 16 is simplified to be

$$\begin{aligned}
\sum_{\tau\in\pi_1\pi_2^{-1}\mathfrak{S}_\gamma}\prod_{k=1}^{r}\prod_{l=k+1}^{r} e^{-cb^{\lambda}_{kl}(\tau)} &= \sum_{\tau\in\pi_1\pi_2^{-1}\mathfrak{S}_\gamma} e^{-cb^{\lambda}_{12}(\tau)} \\
&= \sum_{\tau\in\pi_1\pi_2^{-1}\mathfrak{S}_\gamma} e^{-c(b_{AB}(\tau)+b_{A\bar{B}}(\tau)+b_{\bar{A}B}(\tau)+b_{\bar{A}\bar{B}}(\tau))} \\
&= \sum_{\tau\in\pi_1\pi_2^{-1}\mathfrak{S}_\gamma} e^{-c(b_{AB}(\tau)+0+|\bar{A}||B|+b_{\bar{A}\bar{B}}(\tau))} \\
&= e^{-c|\bar{A}||B|}\sum_{\tau\in\pi_1\pi_2^{-1}\mathfrak{S}_\gamma} e^{-c(b_{AB}(\tau)+b_{\bar{A}\bar{B}}(\tau))} \\
&= e^{-c|\bar{A}||B|}\sum_{\tau\in\mathfrak{S}_t} e^{-cb^{\gamma_1}_{12}(\tau)}\sum_{\tau\in\mathfrak{S}_{n-t}} e^{-cb^{\gamma_2}_{12}(\tau)}
\end{aligned}$$

where $\gamma_1 = (|A|, t-|A|)$ and $\gamma_2 = (|\bar{A}|, n-t-|\bar{A}|)$. The last equality comes from the fact that $\forall\tau \in \pi_1\pi_2^{-1}\mathfrak{S}_\gamma$

$$\begin{aligned}
&\tau^{-1}(i) \in \{1,\ldots,t\} && \text{if} \quad i \in A\cup B, \\
&\tau^{-1}(i) \in \{t+1,\ldots,n\} && \text{if} \quad i \in \bar{A}\cup\bar{B}
\end{aligned}$$

and the choice of representatives π_1, π_2 of the cosets $\mathfrak{S}_\lambda\pi_1, \mathfrak{S}_\gamma\pi_2$ does not change $|A|, |\bar{A}|, |B|$ or $|\bar{B}|$.

By Proposition 11, we have

$$\begin{aligned}
\sum_{\tau\in\mathfrak{S}_t} e^{-cb^{\gamma_1}_{12}(\tau)} &= \frac{|A|!(t-|A|)!\prod_{j=t-|A|+1}^{t}(1-e^{-jc})}{\prod_{j=1}^{|A|}(1-e^{-jc})} \\
\sum_{\tau\in\mathfrak{S}_{n-t}} e^{-cb^{\gamma_2}_{12}(\tau)} &= \frac{|\bar{A}|!(n-t-|\bar{A}|)!\prod_{j=n-t-|\bar{A}|+1}^{n-t}(1-e^{-jc})}{\prod_{j=1}^{|\bar{A}|}(1-e^{-jc})}.
\end{aligned}$$

Substituting the above results into Equation 16, we get

$$\begin{aligned}
&\psi^{-1}(c)|\mathfrak{S}_\gamma|^{-1}\sum_{\sigma\in\mathfrak{S}_\lambda\pi_1}\sum_{\tau\in\mathfrak{S}_\gamma\pi_2} e^{-cd(\sigma,\tau)} \\
&= \frac{\left(e^{-c|\bar{A}||B|}\sum_{\tau\in\mathfrak{S}_t} e^{-cb^{\gamma_1}_{12}(\tau)}\sum_{\tau\in\mathfrak{S}_{n-t}} e^{-cb^{\gamma_2}_{12}(\tau)}\right)\left(\prod_{j=1}^{k}\frac{1-e^{-jc}}{1-e^{-c}}\prod_{j=1}^{n-k}\frac{1-e^{-jc}}{1-e^{-c}}\right)}{\left(\prod_{j=1}^{n}\frac{1-e^{-jc}}{1-e^{-c}}\right)t!(n-t)!} \\
&= \frac{\left(\frac{|A|!(t-|A|)!\prod_{j=t-|A|+1}^{t}(1-e^{-jc})}{\prod_{j=1}^{|A|}(1-e^{-jc})}\right)\left(\frac{|\bar{A}|!(n-t-|\bar{A}|)!\prod_{j=n-t-|\bar{A}|+1}^{n-t}(1-e^{-jc})}{\prod_{j=1}^{|\bar{A}|}(1-e^{-jc})}\right)e^{-c|\bar{A}||B|}\prod_{j=1}^{k}\left(1-e^{-jc}\right)}{t!(n-t)!\prod_{j=n-k+1}^{n}(1-e^{-jc})} \\
&= \frac{|A|!|\bar{A}|!(t-|A|)!e^{-c|\bar{A}||B|}}{t!\prod_{j=n-t-|\bar{A}|+1}^{n-t} j}\left(\frac{\prod_{j=t-|A|+1}^{t}(1-e^{-jc})\prod_{j=n-t-|\bar{A}|+1}^{n-t}(1-e^{-jc})\prod_{j=1}^{k}\left(1-e^{-jc}\right)}{\prod_{j=1}^{|A|}(1-e^{-jc})\prod_{j=1}^{|\bar{A}|}(1-e^{-jc})\prod_{j=n-k+1}^{n}(1-e^{-jc})}\right).
\end{aligned}$$

Note $|A| \le \min(k,t)$, $|\bar{A}| \le k$ and $|B| \le t$, therefore the above expression takes $O(k+t)$ to evaluate. Assuming π_1^{-1} and π_2^{-1} are given, it takes $O(k)$ to get a representative π_1 for the coset $\mathfrak{S}_\lambda\pi_1$, and

$O(t)$ to get the set $\{\pi_1\pi_2^{-1}(1),\cdots,\pi_1\pi_2^{-1}(t)\}$, which completes the proof. ■

Proposition 11 *For $\gamma=(k,n-k)$, let*

$$\mathbf{Q}(k,n) \stackrel{\text{def}}{=} \sum_{\pi\in\mathfrak{S}_n} q^{b_{12}^{\gamma}(\pi)} \tag{20}$$

where $b_{12}^{\gamma}(\pi)$ is defined in Proposition 4, we have

$$\mathbf{Q}(k,n) = k!(n-k)!\frac{\prod_{i=n-k+1}^{n}(1-q^i)}{\prod_{i=1}^{k}(1-q^i)} \quad \forall n\geq k. \tag{21}$$

Proof We first derive an equivalent expression for (20). For fixed π, we sort $\{\pi^{-1}(1),\pi^{-1}(2),\cdots,\pi^{-1}(k)\}$ in ascending order and denote them by $a_1<a_2<\cdots<a_k$. Note that

$$b_{12}^{\gamma}(\pi) = (a_1-1)+(a_2-2)+\cdots+(a_k-k) = \sum_{i=1}^{k}(a_i-i).$$

Due to this observation and since there are $k!(n-k)!$ different permutations for each sequence $(a_1,a_2,\cdots,a_k)$, we have

$$\begin{aligned}\mathbf{Q}(k,n) &= \sum_{\pi\in\mathfrak{S}_n} q^{b_{12}^{\gamma}(\pi)} = k!(n-k)!\sum_{a_k=k}^{n}\sum_{a_{k-1}=k-1}^{a_k-1}\cdots\sum_{a_1=1}^{a_2-1} q^{(a_1+\cdots+a_k-1-\cdots-k)}\\ &= \frac{k!(n-k)!}{q^{\frac{(1+k)k}{2}}}\sum_{a_k=k}^{n} q^{a_k}\sum_{a_{k-1}=k-1}^{a_k-1} q^{a_{k-1}}\cdots\sum_{a_1=1}^{a_2-1} q^{a_1}.\end{aligned} \tag{22}$$

We then prove (21) by mathematical induction on k using the result from (22).

(a) *Base case*: Considering (20) for the case $k=0$ we see that $b_{12}^{(0,n)}(\pi)\equiv 0$ and therefore $\mathbf{Q}(k,n)=n!$ for $n\geq k$. A similar result follows from substituting $k=0$ in the right hand side of (21).

(b) *Inductive step*: Assuming (21) holds $\forall n\geq k$ for some k, we have for $k+1$

$$\begin{aligned}\frac{\mathbf{Q}(k+1,n)}{(k+1)!(n-k-1)!} &\stackrel{a}{=} \frac{1}{q^{\frac{(2+k)(1+k)}{2}}}\sum_{a_{k+1}=k+1}^{n} q^{a_{k+1}}\sum_{a_k=k}^{a_{k+1}-1} q^{a_k}\cdots\sum_{a_1=1}^{a_2-1} q^{a_1}\\ &= \frac{1}{q^{1+k}}\sum_{a_{k+1}=k+1}^{n} q^{a_{k+1}}\left(\frac{1}{q^{\frac{(1+k)k}{2}}}\sum_{a_k=k}^{a_{k+1}-1} q^{a_k}\cdots\sum_{a_1=1}^{a_2-1} q^{a_1}\right)\\ &\stackrel{b}{=} \frac{1}{q^{1+k}}\sum_{a_{k+1}=k+1}^{n} q^{a_{k+1}}\frac{\prod_{i=a_{k+1}-k}^{a_{k+1}-1}(1-q^i)}{\prod_{i=1}^{k}(1-q^i)}\\ &\stackrel{c}{=} \frac{\prod_{i=n-k}^{n}(1-q^i)}{\prod_{i=1}^{k+1}(1-q^i)}\end{aligned}$$

where equality *a* follows from (22), equality *b* follows from the induction hypothesis, and equality *c* follows from Proposition 12.

■

Proposition 12

$$\frac{1}{q^{k+1}} \sum_{j=k+1}^{n} q^j \prod_{i=1}^{k} (1-q^{j-i}) = \frac{1}{1-q^{k+1}} \prod_{i=n-k}^{n} (1-q^i) \qquad \forall n > k.$$

Proof We prove by mathematical induction on n.

(a) *Base Case*: Carefully substituting $n = k+1$ in both the left hand side and the right hand side yields equality.

(b) *Inductive step*:

$$\begin{aligned}
\frac{1}{q^{k+1}} \sum_{j=k+1}^{n+1} q^j \prod_{i=1}^{k} (1-q^{j-i}) &= \frac{1}{q^{k+1}} \left(\sum_{j=k+1}^{n} q^j \prod_{i=1}^{k} (1-q^{j-i}) + q^{n+1} \prod_{i=n-k+1}^{n} (1-q^i) \right) \\
&= \frac{1}{1-q^{k+1}} \prod_{i=n-k}^{n} (1-q^i) + q^{n-k} \prod_{i=n-k+1}^{n} (1-q^i) \\
&= \left(\frac{1-q^{n-k}}{1-q^{k+1}} + q^{n-k} \right) \prod_{i=n-k+1}^{n} (1-q^i) \\
&= \frac{\prod_{i=n-k+1}^{n+1} (1-q^i)}{1-q^{k+1}}
\end{aligned}$$

where in the second equality we used the induction hypothesis.

■

References

W. S. Cleveland. *The Elements of Graphing Data*. Wadsworth Publ. Co., 1985.

D. E. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*. Lecture Notes in Statistics, volume 34, Springer, 1985.

P. Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *IMS Lecture Notes – Monograph Series*. Institute of Mathematical Statistics, 1988.

P. Diaconis. A generalization of spectral analysis with application to ranked data. *Annals of Statistics*, 17(3):949–979, 1989.

M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society B*, 43:359–369, 1986.

M. A. Fligner and J. S. Verducci. Multistage ranking models. *Journal of the American Statistical Association*, 83:892–901, 1988.

M. A. Fligner and J. S. Verducci, editors. *Probability Models and Statistical Analyses for Ranking Data*. Springer-Verlag, 1993.

J. Huang, C. Guestrin, and L. Guibas. Efficient inference for distributions on permutations. In *Advances in Neural Information Processing Systems 20*, pages 697–704. 2008.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30, 1938.

R. Kondor, A. Howard, and T. Jebara. Multi-object tracking with representations of the symmetric group. In *Artificial Intelligence and Statistics*, 2007.

G. Lebanon and J. Lafferty. Conditional models on the ranking poset. In *Advances in Neural Information Processing Systems, 15*, 2003.

C. L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

J. I. Marden. *Analyzing and Modeling Rank Data*. CRC Press, 1996.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the Conference on Computer Supported Cooperative Work*, 1994.

R. P. Stanley. *Enumerative Combinatorics*, volume 1. Cambridge University Press, 2000.

G. L. Thompson. Generalized permutation polytopes and exploratory graphical methods for ranked data. *The Annals of Statistics*, 21(3):1401–1430, 1993.

M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

Journal of Machine Learning Research 9 (2008) 2431-2453 Submitted 12/07; Revised 5/08; Published 11/08

On the Size and Recovery of Submatrices of Ones in a Random Binary Matrix

Xing Sun XING_SUN@MERCK.COM
Merck Research Laboratories
351 N Sumneytown Pike
North Wales, PA 19454-2505, USA

Andrew B. Nobel NOBEL@EMAIL.UNC.EDU
Department of Statistics and Operation Research
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3260, USA

Editor: Nicolas Vayatis

Abstract

Binary matrices, and their associated submatrices of 1s, play a central role in the study of random bipartite graphs and in core data mining problems such as frequent itemset mining (FIM). Motivated by these connections, this paper addresses several statistical questions regarding submatrices of 1s in a random binary matrix with independent Bernoulli entries. We establish a three-point concentration result, and a related probability bound, for the size of the largest square submatrix of 1s in a square Bernoulli matrix, and extend these results to non-square matrices and submatrices with fixed aspect ratios. We then consider the noise sensitivity of frequent itemset mining under a simple binary additive noise model, and show that, even at small noise levels, large blocks of 1s leave behind fragments of only logarithmic size. As a result, standard FIM algorithms, which search only for submatrices of 1s, cannot directly recover such blocks when noise is present. On the positive side, we show that an error-tolerant frequent itemset criterion can recover a submatrix of 1s against a background of 0s plus noise, even when the size of the submatrix of 1s is very small.[1]

Keywords: frequent itemset mining, bipartite graph, biclique, submatrix of 1s, statistical significance

1. Introduction

In many situations, the data obtained from a standard numerical experiment can be represented by a rectangular matrix, whose columns correspond to subjects or samples, and whose rows correspond to variables or features measured for each subject. In a number of important cases, the measured features can take one of two values, and the resulting data can be represented as a binary matrix. Prominent examples include data mining tasks such as frequent pattern mining, single nucleotide polymorphism (SNP) data obtained from inbred strains having two allelic variants, and quantized versions of continuous measurements.

1. A preliminary version of some of the results described here appeared in the work "Significance and Recovery of Block Structures in Binary Matrices with Noise", X. Sun and A.B. Nobel, Proceedings of the 19th Annual Conference on Learning Theory (COLT), H.U. Simon and G. Lugosi eds., Springer, 2006.

The initial analysis of large data sets (typically involving many features and small to moderate numbers of samples) is often exploratory, reflecting the increasing use of such data for hypothesis generation, as well as more traditional hypothesis testing. In unsupervised settings, exploratory analysis seeks to identify patterns or other regularities in the observed data that may point to useful (and potentially unknown) associations between variables, samples or both.

The most common form of exploratory analysis is clustering. Clustering algorithms divide the available samples or variables into disjoint groups so that objects in the same group are, in a suitable sense, close together, while objects in different groups are far apart. A natural extension of standard clustering, usually called biclustering or subspace clustering, looks directly for associations between sets of samples and sets of variables. These associations are represented by submatrices of the data matrix.

In the case of binary matrices, the simplest submatrices of interest are constant, with all entries equal to 1. Submatrices of this sort play a key role in data mining applications, and arise naturally in the study of bipartite graphs (see the discussion below). Motivated in part by these connections, this paper considers the extremal properties of submatrices of 1s in a random binary matrix, and considers the recovery of such submatrices in the presence of noise. More specifically, our analyses are based on a model in which the entries of the principal matrix, and the noise, respectively, are independent Bernoulli(p) random variables. We provide significance bounds for the size of submatrices of 1s under the Bernoulli null hypothesis, and use these to establish limits on the performance of standard data mining methods in the presence of Bernoulli noise. In the same context, we establish several results on the precise asymptotic size of maximal submatrices of 1s, extending to the setting of bipartite graphs earlier work of Bollobás and Erdős (1976) and Matula (1976) on the size of maximal cliques in random graphs. Lastly, we establish finite sample and asymptotic results concerning the recovery of all-1s submatrices in the presence of noise.

1.1 Overview

Connections between binary matrices, frequent itemset mining, and bipartite graphs are discussed in the next section. Section 3 is devoted to the size of the largest square submatrix of 1s in a random binary matrix. Extensions to non-square matrices are described in Section 4. Section 5 contains a short simulation study that supports our theoretical bounds in a non-asymptotic setting. Section 6 is devoted to the noise sensitivity of frequent itemset mining and the recoverability of block structures in the presence of noise.

2. Motivation and Background

An $m \times n$ binary matrix is an indexed family $X = \{x_{i,j} : i \in [m], j \in [n]\}$ where $x_{i,j} \in \{0,1\}$ and $[k]$ denotes the set $\{1,\ldots,k\}$. A submatrix of X is a sub-family $U = \{x_{i,j} : i \in A, j \in B\}$ where $A \subseteq [m]$ and $B \subseteq [n]$; the Cartesian product $C = A \times B$ will be called the index set of U, and we will write $U = X[C]$. When no ambiguity will arise, the index set C itself will be referred to as a submatrix of X.

2.1 Frequent Itemset Mining

Frequent itemset mining (FIM) (Agrawal et al., 1993, 1996), also known as market basket analysis, is a central problem in the field of Data Mining. Generalizations such as bi-clustering and

subspace clustering (Agrawal et al., 1998; Cheng and Church, 2000; Tanay et al., 2002) remain active areas of research. A discussion of FIM and related methods can be found in Hand et al. (2001), Goethals (2003), Madeira and Oliveira (2004) and Tanay et al. (2005).

In the frequent itemset problem, the available data is described by a list $S = \{s_1, \ldots, s_n\}$ of items and a set $T = \{t_1, \ldots, t_m\}$ of transactions. Each transaction t_i consists of a subset of the items in S. If S contains the items available for purchase at a store, then t_i represents a record of the items purchased during the ith transaction, without multiplicity. The goal of FIM is to identify every (maximal) set of items that appear together in more than k transactions, where $k \geq 1$ is a threshold that quantifies "frequent". The data for the FIM problem can readily be represented by an $m \times n$ binary matrix X, with entry $x_{i,j} = 1$ if transaction t_i contains item s_j, and $x_{i,j} = 0$ otherwise. In this form the FIM problem can be stated as follows: given X and $k \geq 1$, find every submatrix of 1s in X having at least k rows, and report the associated set of columns. If the threshold k is allowed to vary, then FIM algorithms essentially seek to find every maximal submatrix of 1s in the data matrix X.

The ongoing application of FIM to large data sets for the purposes of exploratory and related analyses raises a number of natural statistical questions, which we address below in the general setting of random binary matrices. One natural question is how to assign a nominal significance value to the discovery of a moderately sized submatrix of 1s in a large data matrix, accounting for the obvious issue of multiple comparisons arising in this case. Another question is how standard FIM methods perform in the presence of noise, a common feature of many high-throughput measurement technologies. The third question is how one can recover a submatrix of 1s embedded in a larger matrix of 0s when noise is present.

2.2 Bipartite Graphs

Binary matrices are in one to one correspondence with bipartite graphs. An $m \times n$ binary matrix X can be viewed as the adjacency matrix of a graph $G = (V, E)$, where the vertex set V of G is the disjoint union of two sets V_1 and V_2, with $|V_1| = m$ and $|V_2| = n$, corresponding to the rows and columns of X, respectively. There is an edge $(i, j) \in E$ between vertices $i \in V_1$ and $j \in V_2$ if and only if $x_{i,j} = 1$; there are no edges between vertices in V_1 or vertices in V_2. A submatrix U of X with index set $C = A \times B$ corresponds to the subgraph G' of G induced by the vertex set $A \cup B$. If every entry of U is equal to one, then there is an edge (i, j) between every pair of vertices $i \in A$ and $j \in B$, and G' is then a complete bipartite subgraph of G. Thus maximal submatrices of 1s in X correspond to bicliques in G. This connection is the basis for the biclustering algorithm of Tanay et al. (2002).

It is known (cf., Garey and Johnson, 1979; Hochbaum, 1998; Peeters, 2003) that the problem of finding a biclique with the largest number of edges in a given bipartite graph G is NP-complete, and thus the same is true of the general frequent itemset problem with no restriction on the threshold k. Several approximate methods (Hochbaum, 1998; Dawande et al., 2001; Mishra et al., 2004) have been proposed for finding large bicliques in bipartite graphs in polynomial time. Mishra et al. (2004) show that the results provided by their randomized algorithm overlap a large fraction of the largest bicliques with high probability.

Our interest here is in assessing the significance and extremal size of maximal bicliques in random bipartite graphs. We do not address the question of how to search for such bicliques, and refer the interested reader to the papers above and the references therein for more details.

3. Largest Submatrices of 1s: Square Case

In this section we study the size of the largest square submatrix of 1s in a square binary matrix whose entries are independent Bernoulli(p) random variables. Non-square matrices and submatrices are considered in Section 4.

Definition: Let $Z = \{z_{i,j} : i, j \geq 1\}$ be an infinite array of independent binary random variables with $P(z_{i,j} = 1) = p = 1 - P(z_{i,j} = 0)$, where the probability $p \in (0,1)$ is fixed. For $n \geq 1$, let $Z_n = \{z_{i,j} : 1 \leq i, j \leq n\}$.

Thus Z_n is an $n \times n$ binary random matrix comprising the "upper left corner" of the collection $\{z_{i,j}\}$. This definition allows us to make almost-sure type statements concerning the asymptotic behavior of functions of Z_n.

Definition: Given a binary matrix X, let $M(X)$ be the largest k such that there exists a $k \times k$ submatrix of 1s in X. Note that $M(X)$ is invariant under row and column permutations of X.

From a statistical point of view, the random matrix Z_n follows a simple null model under which the observed binary data matrix has no special structure, and $M(\cdot)$ acts as a natural test statistic with which to detect departures from the null. Our analysis begins with a bound on the probability that $M(Z_n)$ exceeds a fixed integer $k \geq 1$. We follow a standard first moment argument (cf., Alon and Spencer, 1991).

Fix n for the moment, and for each $1 \leq k \leq n$ let U_k be the number of $k \times k$ submatrices of ones in Z_n. Then, letting $S = \{C = A \times B : A, B \subseteq [n], |A| = |B| = k\}$, we may write

$$U_k \;=\; \sum_{C \in S} I\{\text{all entries of } Z_n[C] \text{ are } 1\}$$

from which it follows that

$$EU_k \;=\; |S| \cdot P(\text{all entries of } Z_n[C] \text{ are } 1) \;=\; \binom{n}{k}^2 p^{k^2}.$$

By Markov's inequality and the previous display,

$$P\,(M(Z_n) \geq k) \;=\; P(U_k \geq 1) \;\leq\; EU_k \;=\; \binom{n}{k}^2 p^{k^2}. \tag{1}$$

We wish to identify an integer k_n for which EU_{k_n} is approximately equal to one. For values $k > k_n$ the rightmost expression in (1) provides an effective means for bounding the probability on the left. Note that $EU_n = p^{n^2} < 1$, and $EU_1 = n^2 p > 1$ when n is sufficiently large. Moreover, it is clear from the definition that $U_{k+1} \leq U_k$, so that EU_k is non-increasing in k. Using the Stirling approximation of the rightmost expression in (1), define

$$\phi_n(s) \;=\; (2\pi)^{-\frac{1}{2}}\, n^{n+\frac{1}{2}}\, s^{-s-\frac{1}{2}}\, (n-s)^{-(n-s)-\frac{1}{2}}\, p^{\frac{s^2}{2}}, \qquad s \in (0,n).$$

The quantity $\phi_n(k)$ is an approximation of $(EU_k)^{1/2}$: the ratio $\phi_n(k)/(EU_k)^{1/2}$ is bounded away from zero and infinity, independent of n, k, and tends to one if k and $n-k$ tend to infinity with n. Let $s(n)$ be any positive real root of the equation

$$1 = \phi_n(s). \tag{2}$$

The next lemma shows that $s(n)$ is unique and grows as logarithmically with n.

Lemma 1. *When n is sufficiently large, the Equation (2) has a unique root $s(n)$ satisfying $\log_b n < s(n) < 2\log_b n$, where $b = p^{-1}$.*

Using the bounds of Lemma 1 and some technical but straightforward calculations, one may obtain a simple asymptotic expression for $s(n)$.

Lemma 2. *The root $s(n)$ defined by (2) has the form*

$$s(n) \;=\; 2\log_b n - 2\log_b \log_b n + C + o(1)$$

where $b = p^{-1}$ and $C = 2\log_b e - 2\log_b 2$.

The proofs of Lemmas 1 and 2 can be found in Section 7.1. Let $k(n) = \lceil s(n) \rceil$ be the least integer greater than or equal to $s(n)$. The next proposition provides an upper bound on $P(M(Z_n) \geq k)$ for $k > k(n)$. Its proof appears in Section 7.2.

Proposition 1. *For each $\varepsilon > 0$, when n is sufficiently large, $P\left(M(Z_n) \geq k(n) + r\right) \leq n^{-2r} \left(\log_b n\right)^{2r+\varepsilon}$.*

One may obtain a cruder bound, on the probability that $M(Z_n)$ is at least $2\log_b n + r$, in a simpler fashion by noting that

$$EU_k = \binom{n}{k}^2 p^{k^2} \leq \frac{n^{2k}}{k!^2} e^{-k^2 \log b} \leq \frac{e^{2k\ln n - k^2 \ln b}}{k^2} \leq n^{-2r}$$

when $k \geq 2\log_b n + r$. Both the upper bound of Proposition 1 and the definition of $s(n)$ are based on the inequality (1), which follows from a simple union bound on the probability that $M(Z_n)$ is at least k. The union bound is typically quite loose, but it is sufficiently strong in this context to ensure that, for large n, the random variable $M(Z_n)$ is close to the threshold $s(n)$. Indeed, it follows from Proposition 1 and the first Borel Cantelli Lemma that, with probability one, $M(Z_n)$ is eventually less than $s(n) + 1$. Using a more involved second moment argument, one can establish a corresponding lower bound on $M(Z_n)$. Together these bounds yield the following result.

Theorem 1. *Given any $\varepsilon > 0$, with probability one, $s(n) - 1 - \varepsilon < M(Z_n) < s(n) + \varepsilon$ when n is sufficiently large.*

It follows from Theorem 1 that for large n the size of the largest square submatrix of 1s in Z_n can take one of at most two integer values in an interval of width $1 + 2\varepsilon$ containing the number $s(n)$. Indeed, it is shown in the proof of Theorem 1 that there is a sequence of integers $\{r(n)\}$ close to $\{s(n)\}$ such that, with probability one, when n is sufficiently large $M(Z_n) \in \{r_n - 1, r(n)\}$. Thus $M(Z_n)$ exhibits two-point concentration and does not possess a limiting continuous distribution.

The proof of Theorem 1 is given in Section 8. The outline of the proof follows arguments of Bollobás and Erdős (1976), who studied the size of the largest clique $cl(G_n)$ in a random graph G_n with n vertices, where each edge is included independently with probability p. They showed that for a deterministic function $c(n)$, equal to $s(n)$ up to the constant and lower order terms, eventually almost surely $|cl(G_n) - c(n)| < 3/2$. Matula (1976) independently established a similar result. See these references or Bollobás (2001) for more details. Theorem 1 extends these results to balanced

bicliques in balanced bipartite random graphs. (Unbalanced bipartite graphs are considered in the next section.)

Dawande et al. (2001) used first and second moment arguments to show (in our terminology) that $P(\log_b n \leq M(Z_n) \leq 2\log_b n) \to 1$ as n tends to infinity. Improving these results, Park and Szpankowski (2005) showed that $P((1+\varepsilon)\log_b n \leq M(Z_n) \leq (2-\varepsilon)\log_b n)$ tends to 1 as n tends to infinity for any fixed $0 < \varepsilon < 1$. Koyutürk et al. (2004) studied the problem of finding dense patterns in binary data matrices. They used a Chernoff type bound for the binomial distribution to assess whether an individual submatrix has an enriched fraction of ones, and employed the resulting test as the basis for a heuristic search for significant bi-clusters. However, the effects of multiple testing are not considered in their assessments of significance. Tanay et al. (2002) assessed the significance of bi-clusters in a real-valued matrix using likelihood-based weights, a normal approximation and a standard Bonferroni correction to account for the multiplicity of submatrices. Use of the normal approximation for individual submatrices leads to subtoptimal bounds in non-Gaussian settings.

3.1 Smallest Maximal Submatrix of 1s

Square submatrices of 1s will occur by chance in a random binary matrix. The largest such submatrix has approximately $2\log_b n - 2\log_b \log_b n$ rows. Conversely, one may ask about the size of the *smallest* maximal square submatrix of 1s. (A square submatrix of 1s is maximal if there is no larger square submatrix of 1s that properly contains it.)

Definition: Let $L(Z_n)$ be the smallest k such that there exists at least one $k \times k$ maximal submatrix of 1's in Z_n.

Theorem 1 implies that $L(Z_n) \leq 2\log_b n$. An analysis based on second moment arguments similar to those used in the proof of Theorem 1 yields the following, tighter bound. The proof can be found in Sun (2007).

Theorem 2. *With probability one,*

$$\lim_{n\to\infty} \frac{L(Z_n)}{\log_b n} = 1.$$

Bollobás and Erdős (1976) establish a related result on the size of the smallest clique in a random graph. However their proof can not be directly extended to obtain the theorem above. Indeed, an extension of their argument provides a lower bound on the size of the smallest square submatrix of 1s that is not properly contained within a rectangular submatrix of 1s, and the resulting bound is necessarily larger than the one in Theorem 2.

4. Non-Square Matrices

In this section we consider the case where the primary matrix and the target submatrices of 1s may be rectangular, but maintain fixed row/column aspect ratios as the size of the primary matrix grows. Natural analogs of Proposition 1 and Theorem 1 are obtained in this setting. For $m, n \geq 1$ define the random matrix $Z(m,n) = \{z_{i,j} : i \in [m], j \in [n]\}$.

Definition: Let $\alpha > 0$ and $\beta > 0$ be aspect ratios for the primary matrix and target submatrices, respectively. Define $M_n(Z : \alpha, \beta)$ to be the largest k such that $Z(\lceil \alpha n \rceil, n)$ contains a $\lceil \beta k \rceil \times k$ submatrix of 1s.

The asymptotic behavior of $M_n(Z:\alpha,\beta)$ is the same as that of $M_n(Z:\alpha^{-1},\beta^{-1})$, so we assume in what follows that $\beta \geq 1$. The analysis of $M_n(Z:\alpha,\beta)$ proceeds along the same lines as that of $M(Z_n)$. Investigating the value of k for which the expected number of $\lceil \beta k \rceil \times k$ submatrices of 1s in $Z(\lceil \alpha n \rceil, n)$ is equal to 1, we arrive at the function

$$s(n,\alpha,\beta) \;=\; \frac{1+\beta}{\beta}\log_b n - \frac{1+\beta}{\beta}\log_b\left(\frac{1+\beta}{\beta}\log_b n\right) + \log_b \alpha + C(\beta) + o(1),$$

where $b = p^{-1}$ and $C(\beta) = \beta^{-1}((1+\beta)\log_b e - \beta \log_b \beta)$ depends only on β.

Note that the aspect ratio α of the primary matrix appears only in the constant term of $s(n,\alpha,\beta)$, and therefore plays only a minor role in what follows. The proofs of Proposition 2 and Theorem 3 below are similar to their analogs in the square case, with additional notation and work required to handle the two aspect ratios, and are omitted. Detailed arguments can be found in Sun (2007).

Proposition 2. *Fix aspect ratios $\alpha > 0$, $\beta \geq 1$. For every $\varepsilon > 0$, when n is sufficiently large* $P(M_n(Z:\alpha,\beta) \geq \lceil s(n,\alpha,\beta) \rceil + r) \;\leq\; n^{-(\beta+1)r}(\log_b n)^{(\beta+1+\varepsilon)r}$.

Remark: When the aspect ratio α of the primary matrix is fixed, it does not play an essential role in the asymptotic behavior of $M_n(Z:\alpha,\beta)$, which is dominated by higher order factors involving only the aspect ratio β of the target submatrices. It is natural then to consider a situation in which the aspect ratio α of the primary matrix can increase with n. This might model, for example, the scaling and cost structure of a given high-throughput technology over time. In the case where $\alpha(n) = n^{\gamma}$ for some $\gamma > 0$, the proof of Proposition 2 can be modified to show that

$$P\left(M_n(Z:n^{\gamma},\beta) \geq \left(\gamma + \frac{\beta+1}{\beta}\right)\log_b n\right) \leq n^{-(\beta+1)r}(\log_b n)^{(\beta+1+\varepsilon)r}.$$

On the other hand, one can readily show that if $\beta \geq 1$ is fixed and m grows exponentially with n, then $Z(m,n)$ will contain a $\lceil \beta n \rceil \times n$ submatrix of 1's with probability bounded away from zero. For fixed aspect ratios α and β one may obtain an asymptotic concentration result for $M_n(Z:\alpha,\beta)$ analogous to Theorem 1.

Theorem 3. *For fixed $\alpha > 0$ and $\beta \geq 1$, with probability one $|M_n(Z:\alpha,\beta) - s(n,\alpha,\beta)| \leq \frac{5}{2}$ when n is sufficiently large.*

Theorem 3 implies that $Z(\alpha n, n)$ contains a submatrix of 1s having aspect ratio β and area $(\beta+1)\log_b^2 n$, the latter increasing with β. Park and Szpankowski (2005) establish a related result, showing that if we do not restrict β, the aspect ratio of the submatrices, then with high probability the submatrix of 1s in $Z(m,n)$ with the largest area is of size $O(n) \times \ln b$ or $\ln b \times O(n)$.

5. Simulation Study

The results of the previous sections hold when n is sufficiently large. In order to assess their validity for moderate values of n, we carried out a simple simulation study. For $n = 40$ and $n = 80$ we generated 400 $n \times n$ random binary matrices with $p = .2$, $p = .3$ and $p = .35$ respectively. Then we applied the FP-growth algorithm (Han et al., 2000) to identify all maximal submatrices of ones. For each maximal submatrix of ones we recorded the length of its shorter side, and let $\hat{M}$ be the maximum among these lengths. Thus $\hat{M}$ is the side length of the largest square submatrix of 1's in

p	n	$s(n)$	k	Proportion of $\hat{M}=k$
0.2	40	3.55	3	85.75%
			4	14.25%
	80	4.58	4	97%
			5	3%
0.3	40	4.78	4	50.5%
			5	49.5%
	80	5.64	5	85%
			6	15%
0.35	40	5.22	4	63.75%
			5	36%
			6	0.25%
	80	6.21	5	7.75%
			6	90.75%
			7	1.50%

Table 1: Distribution of observed $\hat{M}(Z_n)$ based on simulation

the generated random matrix. We recorded the values of $\hat{M}$ over all simulations and compared these values to the corresponding bounds. Table 1 summarizes the results. Note that in each simulation $-1.5 < \hat{M} - s(n) < 1$.

In order to check the theoretical bounds on $M_n(Z:1,\beta)$ with $\beta \geq 1$, we considered the 400 random 80×80 matrices with p=0.3 used to evaluate the result for square submatrices above. For each such matrix, we identified all maximal rectangular submatrices of 1s, and recorded the length of both their longer and shorter sides. For each $\beta \geq 1$ we defined $\hat{M}(\beta)$ to be the largest k such that at least one $\lceil \beta k \rceil \times k$ or $k \times \lceil \beta k \rceil$ submatrix of 1's was observed. The difference between $\hat{M}(\beta)$ and $s(80,1,\beta)$ was calculated and is displayed in Figure 1. The x-axes in both panels are equal to $1/\beta$. The y-axis in the left panel is the difference between $\hat{M}(\beta)$ and $s(80,1,\beta)$, and the y-axis in the right panel is the proportion of simulations which are inconsistent with the theoretical predictions of Theorem 3. Note that even for the moderate matrix size $n = 80$, the theoretical predictions are very accurate when the aspect ratio β is less than 2.5. In these cases, all the observed size lengths are within the range of predicted values.

6. Fragmentation and Recovery in the Presence of Noise

In this section we shift our attention from submatrices of 1s in Z_n to a setting in which Z_n plays the role of binary noise. Formally, we study the additive model

$$Y_n \;=\; X_n \oplus Z_n, \tag{3}$$

where each matrix is of dimension $n \times n$. The operation $\oplus$ is the standard exclusive-or: $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$. The matrix $X_n = \{x_{i,j}\}$ is a non-random binary matrix that contains the "true" values of interest, in the absence of noise, and Z_n is a random binary matrix that acts as noise, with intensity $p \in (0,1)$. The matrix $Y_n = \{y_{i,j} = x_{i,j} \oplus z_{i,j}\}$ represents the observed binary

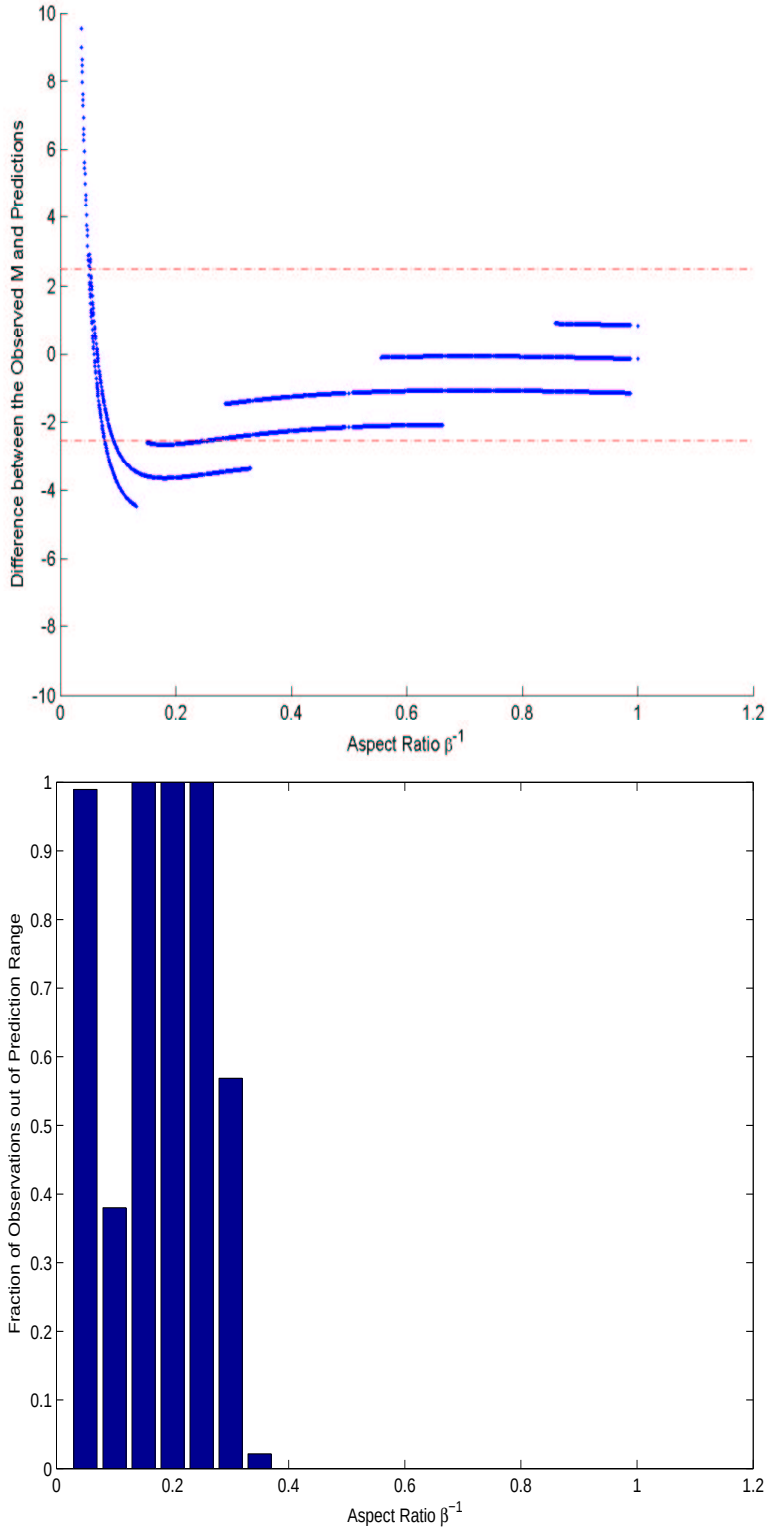


Figure 1: Difference between observed $\hat{M}(\beta)$ and its predicted value from theory.

data. Thus the effect of the noise is to randomly flip some of the values of X in Y. The model (3) is the binary version of the standard additive noise model common in statistical inference.

6.1 Noise Sensitivity

Much of the data to which data mining methods are applied is obtained by high-throughput technologies or the automated collection of information from diverse sources with varying levels of reliability. The resulting data sets are often subject to moderate levels of error and noise. Noise can also arise when binary data are obtained by thresholding continuous data, as is sometimes done in microarray analyses. Whatever its source, noise can potentially have serious consequences for frequent itemset methods if they are applied in a direct way to identify submatrices of 1s.

Indeed, this conclusion is already apparent from Theorem 1. If each entry of the target matrix X_n is zero, then $Y_n = Z_n$ and the largest $k \times k$ submatrix of ones in Y_n has $k \approx 2\log_b n$ with $b = p^{-1}$. At the other extreme, if every entry of X_n is equal to one, then the entries of Y_n are independent Bernoulli$(1-p)$ random variables, and in this case the largest square submatrix of ones in Y has side-length $k \approx 2\log_{b'} n$ with $b' = (1-p)^{-1}$. The next result extends this reasoning to any underlying target matrix X_n.

Proposition 3. *Fix $0 < p < 1/2$. Let $\{X_n\}$ be any sequence of $n \times n$ square binary matrices, and let $Y_n = X_n \oplus Z_n$. For each $\varepsilon > 0$, eventually almost surely $(2-\varepsilon)\log_b n < M(Y_n) \leq 2\log_{b'} n$, where $b = p^{-1}$ and $b' = (1-p)^{-1}$.*

Proof of Proposition 3: Fix n and let $\tilde{W}_n = \{\tilde{w}_{i,j}\}$ be an $n \times n$ binary matrix with independent entries, defined on the same probability space as $\{z_{i,j}\}$, such that

$$\tilde{w}_{i,j} = \begin{cases} \text{Bern}\left(\frac{1-2p}{1-p}\right) & \text{if } x_{ij} = y_{ij} = 0 \\ 1 & \text{if } x_{ij} = 0, y_{ij} = 1 \\ y_{i,j} & \text{if } x_{ij} = 1 \end{cases}$$

where the Bernoulli variable in the first condition is independent of $\{z_{i,j}\}$. Define $\tilde{Y}_n = Y_n \vee \tilde{W}_n$ to be the entry-wise maximum of Y_n and $\tilde{W}_n$. Then clearly $M(Y_n) \leq M(\tilde{Y}_n)$, as any submatrix of ones in Y_n must also be present in $\tilde{Y}_n$. Moreover, the variables $\tilde{y}_{i,j}$ are i.i.d. with $P(\tilde{y}_{i,j} = 1) = 1-p$, so that we may regard $\tilde{Y}_n$ as a Bern$(1-p)$ noise matrix. It then follows from Theorem 1 that $M(Y_n) \leq 2\log_{b'} n$ eventually almost surely. To obtain the other inequality, define

$$\hat{w}_{i,j} = \begin{cases} \text{Bern}\left(\frac{p}{1-p}\right) & \text{if } x_{ij} = y_{ij} = 1 \\ 0 & \text{if } x_{ij} = 1, y_{ij} = 0 \\ y_{i,j} & \text{if } x_{ij} = 0 \end{cases}$$

and let $\hat{Y}_n = Y_n \wedge \hat{W}_n$ be the entry-wise maximum of Y_n and $\hat{W}_n$. It is easy to verify that $M(Y_n) \geq M(\hat{Y}_n)$, and that the entries in $\hat{Y}_n$ are i.i.d. Bern(p). Theorem 1 then implies that $M(Y_n) \geq (2-\varepsilon)\log_b n$ eventually almost surely.

Proposition 3 can be interpreted as follows. No matter what type of block structures might exist in X, in the presence of random noise these structures leave behind only logarithmic fragments in the observed data. Under the additive noise model (3), block structures in X cannot be recovered directly by methods such as frequent itemset mining that look for maximal submatrices of ones without errors.

6.2 Recovery

In light of Proposition 3, it is natural to consider methods for identifying submatrices of 1s that may be contaminated with a certain fraction of 0s. These submatrices correspond, in the data mining and bipartite graph settings, to approximate frequent itemsets and approximate bicliques, respectively. A number of different error-tolerant frequent itemset mining algorithms have been proposed in the literature (Pei et al., 2001, 2002; Yang et al., 2001; Seppänen and Mannila, 2004; Liu et al., 2005, 2006). Most are based on criteria that require the average of the identified submatrices to be greater than a user specified threshold τ. One can readily adapt the first moment argument to obtain significance bounds for submatrices with a large fraction of 1s; details can be found in Sun (2007).

Here we consider the simple problem of recovering a (potentially small) submatrix C of 1s embedded in a matrix of 0s from a single noisy observation. Proposition 3 shows that one cannot recover C directly using standard frequent itemset mining; instead we consider the Approximate Frequent Itemset (AFI) algorithm developed in Liu et al. (2005).

Definition: Given a binary matrix U with index set C, let $F(U) = |C|^{-1} \sum_{(i,j)\in C} u_{i,j}$ be the fraction of ones in U, or equivalently, the average of the entries of U.

Let u_{i*} and u_{*j} denote the rows and columns, respectively, of a given submatrix U.

Definition: Let $\tau \in [0,1]$ be fixed. A submatrix U of a binary matrix Y is a τ-*approximate frequent itemset* (τ-AFI) if each of its rows satisfies $F(u_{i*}) \geq \tau$ and each of its columns satisfies $F(u_{*j}) \geq \tau$. Define $\mathrm{AFI}_\tau(Y)$ to be the collection of all τ-AFIs in Y.

The definition above comes from Liu et al. (2005), who presented an algorithm for identifying AFIs in binary matrices.

Let X_n be an $n \times n$ binary matrix that consists of an $l \times l$ submatrix of ones having index set C^*, with all other entries equal to 0. (The rows and columns of C^* need not be contiguous.) Suppose that $Y_n = X_n \oplus Z_n$, where Z_n has noise level $p \in (0, 1/2)$. We wish to recover the index set C^* of the target submatrix from Y_n.

To this end, assume that the noise level p is unknown, but that there is a known upper bound p_0 such that $p < p_0 < 1/2$, and let $\tau = 1 - p_0$ be an associated error threshold. We estimate C^* by the index set of the largest square τ-AFI in the observed matrix Y_n. More precisely, let $\hat{\mathscr{C}}$ be the family of index sets of square submatrices $U \in \mathrm{AFI}_\tau(Y_n)$, and let

$$\hat{C} = \operatorname*{argmax}_{C \in \hat{\mathscr{C}}} |C|$$

be the index set of any maximal sized submatrix in $\hat{\mathscr{C}}$. (The set $\hat{\mathscr{C}}$ contains 1×1 submatrices with entry 1, so it is non-empty whenever Y_n is not identically 0.) Note that $\hat{\mathscr{C}}$ and $\hat{C}$ depend only on the observed matrix Y_n. Let the ratio

$$\Lambda = |\hat{C} \cap C^*| / |\hat{C} \cup C^*|$$

measure the overlap between the estimated index set $\hat{C}$ and the true index set C^*. Clearly $0 \leq \Lambda \leq 1$, and values of Λ close to one indicate better overlap. The proof of the next theorem is given in Section 9.

Theorem 4. *When n is sufficiently large, for any* $0 < \alpha < 1$ *such that* $8\alpha^{-1}(\log_b n + 2) \le l$ *we have*

$$P\left(\Lambda \le \frac{1-\alpha}{1+\alpha}\right) \le \Delta_1(l) + \Delta_2(\alpha, l).$$

Here $\Delta_1(l) = 2le^{-\frac{3l(p-p_0)^2}{8p}}$ *and* $\Delta_2(\alpha, l) = 2n^{-\frac{1}{4}\alpha l + 2\log_b n}$, *with* $b = \exp\{3(1-2p_0)^2/8p\}$.

Remarks: The second term $\Delta_2(\alpha, l)$ is less than $2n^{-4/\alpha}$ and is the dominant term in the probability upper bound if $l/\ln(n)$ is large. The logarithmic base b is derived from an upper bound on the tails of the binomial distribution, and is always larger than $\tilde{b} = \exp\{3(1-2p_0)^2/8p_0\}$. By a crude bound, $\Delta_1(l) \le \tilde{\Delta}_1(l) := e^{-\sqrt{l}}$ when l is sufficiently large. Thus, by replacing b with $\tilde{b}$ and $\Delta_1(l)$ with $\tilde{\Delta}_1(l)$, one obtains a probability bound that does not depend on the unknown parameter p.

As a corollary of Theorem 4, we can also get results in an asymptotic setting. Suppose that $\{X_n : n \ge 1\}$ is a sequence of square binary matrices, and that X_n contains an $l_n \times l_n$ submatrix C_n^* of 1s with all other entries equal to 0. Let $Y_n = X_n \oplus Z_n$, and let Λ_n measure the overlap between C_n^* and the estimate $\hat{C}_n$ produced by the AFI-based recovery method above. The following result follows from Theorem 4 and the Borel Cantelli lemma.

Corollary 1. *If* $l_n \ge 8\psi(n)(\log_b n + 2)$ *where* $\psi(n) \to \infty$ *as* $n \to \infty$, *then eventually almost surely*

$$\Lambda_n \ge \frac{1-\psi(n)^{-1}}{1+\psi(n)^{-1}} \to 1.$$

Reuning-Scherer studied several recovery problems in his thesis (Reuning-Scherer, 1997). In the case considered here, he calculated the fraction of 1s in every row and every column of Y, and then selected those rows and columns for which these fractions exceeded an appropriate threshold. His algorithm is easily seen to be consistent when $l \ge n^\alpha$ for $\alpha > 1/2$. However, it is easy to show using the central limit theorem that individual row and column sums alone are not sufficient to recover C^* when $l \le n^\alpha$ for $\alpha < 1/2$. In the latter case, one gains considerable power by directly considering submatrices, and as the result above demonstrates, one can consistently recover C_n^* if $l_n/\ln(n) \to \infty$.

7. Proofs of Preliminary Results

In this section, we will begin with the proofs of Lemma 1 and Lemma 2 then follow with the proof of Proposition 1.

7.1 Proofs of Lemmas 1 and 2

Proof of Lemma 1: Differentiating $\log_b(\phi_n(s))$ yields

$$\frac{\partial \log_b(\phi_n(s))}{\partial s} = \frac{1}{2(n-s)\ln b} + \log_b(n-s) - s - \log_b s - \frac{1}{2s\ln b},$$

which is negative when $\log_b n < s < 2\log_b n$. A routine calculation shows that for $0 < s \le \log_b n$,

$$\begin{aligned}\log_b \phi_n(s) &= (n+\frac{1}{2})\log_b n - (s+\frac{1}{2})\log_b s - (n-s+\frac{1}{2})\log_b(n-s) - \frac{s^2}{2} - \frac{1}{2}\log_b 2\pi \\ &\ge s\left(\log_b(n-\log_b n) - \frac{s}{2} - \log_b \log_b n\right) - \frac{1}{2}\log_b s - \frac{1}{2}\log_b 2\pi \ > 0\end{aligned}$$

when n is sufficiently large. Similarly, for $2\log_b n \le s < n$,

$$\begin{aligned}\log_b \phi_n(s) &\le s\left(\log_b(n-s) - \frac{s}{2} - \log_b s\right) - \frac{1}{2}\log_b s - \frac{1}{2}\log_b 2\pi + 2s + \frac{s\log_b s}{2} \\ &\le s\left(2 - \frac{\log_b s}{2}\right) - \frac{1}{2}\log_b s - \frac{1}{2}\log_b 2\pi \;<\; 0\end{aligned}$$

when n is sufficiently large. Thus for sufficiently large n, there exists a unique solution $s(n)$ of the equation $\phi_n(s) = 1$ with $s(n) \in (\log_b n, 2\log_b n)$. ■

Proof of Lemma 2: Taking logarithms of both sides of the equation $\phi_n(s) = 1$ and rearranging terms yields

$$\frac{1}{2}\log_b \frac{n}{n-s} + n\log_b \frac{n}{n-s} - (s+\frac{1}{2})\log_b s + s\log_b(n-s) - \frac{s^2}{2} = \frac{\log_b 2\pi}{2}.$$

Lemma 1 implies that $s(n)$ belongs to the interval $(\log_b n, 2\log_b n)$, so we consider the above equation in the case that $n >> s$. Dividing both sides of the equation by s yields

$$\log_b(n-s) - \frac{s}{2} - \log_b s = -\log_b e + O(\frac{\log_b s}{s}),$$

which can be rewritten as

$$\log_b n - \frac{s}{2} - \log_b \log_b n = \log_b \frac{s}{\log_b n} - \log_b \frac{n-s}{n} - \log_b e + O(\frac{\log_b s}{s}). \tag{4}$$

For each n, define $R(n)$ via the equation

$$s(n) \;=\; 2\log_b n - 2\log_b \log_b n + R(n).$$

Plugging this expression into (4), it follows that $R(n) = 2\log_b e - 2\log_b 2 + o(1)$, and the result follows from the uniqueness of $s(n)$. ■

7.2 Proof of Proposition 1

To establish the bound with r independent of n, it suffices to consider a sequence r_n that changes with n in such a way that $1 \le r_n \le n$. Fix n for the moment, let $l = k(n) + r_n$, and let U_l be the number of $l \times l$ submatrices of 1s in Z_n. Then by Markov's inequality and Stirling's approximation,

$$P(M(Z_n) \ge l) \;=\; P(U_l \ge 1) \;\le\; E(U_l) \;=\; \binom{n}{l}^2 p^{l^2} \;\le\; 2\phi_n^2(l).$$

A straightforward calculation using the definition of $\phi_n(\cdot)$ shows that one can decompose the rightmost term above as follows:

$$2\phi_n^2(l) \;=\; 2\,\phi_n^2(k(n))\, p^{r\cdot k(n)}\, [A_n(r)\, B_n(r)\, C_n(r)\, D_n(r)\,]^2,$$

where

$$A_n(r) = \left(\frac{n-r-k(n)}{n-k(n)}\right)^{-n+r+k(n)+\frac{1}{2}}, \quad B_n(r) = \left(\frac{r+k(n)}{k(n)}\right)^{-k(n)-\frac{1}{2}},$$

$$C_n(r) = \left(\frac{n-k(n)}{r+k(n)}\, p^{\frac{k(n)}{2}}\right)^{r}, \quad D_n(r) = p^{\frac{r^2}{2}}.$$

Note that $p^{r\cdot k(n)} = o(n^{-2r}(\log_b n)^{2r+\varepsilon})$ for any fixed $\varepsilon > 0$, and that $\phi_n^2(k(n)) \le 1$ by the monotonicity of $\phi_n(\cdot)$ and the definition of $k(n)$. Thus it suffices to show that $A_n(r)\cdot B_n(r)\cdot C_n(r)\cdot D_n(r) = O(1)$ when n is sufficiently large. To begin, note that for any fixed $\delta \in (0,1/2)$, when n is sufficiently large,

$$C_n(r)^{\frac{1}{r}} = \frac{n-k(n)}{r+k(n)}\, p^{\frac{k(n)}{2}} \le \frac{n}{k(n)}\, p^{\frac{k(n)}{2}} \le \frac{n}{(2-\delta)\log_b n}\frac{\frac{2+\delta}{2}\log_b n}{n},$$

which is less than one. Note that $B_n(r) \le 1$. It only remains to show $A_n(r)\cdot D_n(r) = O(1)$. Simple calculations yield that $\ln A_n(r) \le r$. Consequently, $\ln(A_n(r)\cdot D_n(r)) \le r - \frac{r^2 \ln b}{2}$, which is bounded from above. ■

8. Proof of Theorem 1

The proof of Theorem 1 is established via a sequence of technical lemmas. Modifying our earlier notation slightly, let $U_k(n)$ denote the number of $k\times k$ submatrices of 1s in Z_n. In what follows ε is a fixed positive number less than $\frac{1}{2}$. Our argument parallels that outlined in Bollobás (2001). We begin with the following definition.

Definition: For each $k \ge 1$, let n_k' be the least integer n such that

$$EU_k(n) \ge k^{3+\varepsilon},$$

and let n_k be the largest integer n such that

$$EU_k(n) \le k^{-3-\varepsilon}.$$

Note that n_k and n_k' exist for sufficiently large $k \ge 1$, as $EU_k(k) = p^{k^2} \le k^{-3-\varepsilon}$, $EU_k(n)$ is monotone increasing in n, and $EU_k(n) \to \infty$ as $n \to \infty$.

Lemma 3. *Let n_k and n_k' be defined as above.*

a. When k is sufficiently large, $n_k' < n_{k+1}$.

b. When k is sufficiently large, $n_k' - n_k < C_1 \frac{n_k \ln k}{k}$ for some constant $C_1 > 2$.

c. $\lim_{k\to\infty} \frac{n_{k+2}-n_{k+1}}{n_{k+1}-n_k} = b^{\frac{1}{2}}$.

Proof of (a): It follows from the definition of n_k that

$$\binom{n_k}{k} p^{\frac{k^2}{2}} \le k^{-\frac{(3+\varepsilon)}{2}} \quad \text{and} \quad \binom{n_k+1}{k} p^{\frac{k^2}{2}} \ge k^{-\frac{(3+\varepsilon)}{2}}. \tag{5}$$

Rearranging terms in the first inequality, and noting that $(n_k-k)!/n_k! \leq (n_k-k)^{-k}$ we obtain, in turn, the inequalities

$$\frac{k^{\frac{(3+\varepsilon)}{2}}}{k!\, b^{\frac{k^2}{2}}} \leq \frac{1}{(n_k-k)^k} \quad \text{and} \quad n_k \leq b^{\frac{k}{2}} \left[\frac{k!}{k^{\frac{(3+\varepsilon)}{2}}}\right]^{\frac{1}{k}} + k.$$

Rearranging the terms in the second inequality of (5), one may establish by a similar argument the inequalities

$$k^{\frac{(3+\varepsilon)}{2}} \geq b^{\frac{k^2}{2}} \frac{k!}{(n+1)^k} \quad \text{and} \quad n_k \geq b^{\frac{k}{2}} \left(\frac{k!}{k^{\frac{3+\varepsilon}{2}}}\right)^{\frac{1}{k}} - 1.$$

Combining the two bounds on n_k above, yields

$$b^{\frac{k}{2}} \left(k!\, k^{-\frac{3+\varepsilon}{2}}\right)^{\frac{1}{k}} - 1 \leq n_k \leq b^{\frac{k}{2}} \left(k!\, k^{-\frac{(3+\varepsilon)}{2}}\right)^{\frac{1}{k}} + k \tag{6}$$

and the asymptotic relation

$$n_k = b^{\frac{k}{2}} (k!)^{\frac{1}{k}} + o(k b^{\frac{k}{2}}). \tag{7}$$

From the definition of n'_k, one can establish in a similar fashion the inequalities

$$b^{\frac{k}{2}} \left(k!\, k^{\frac{3+\varepsilon}{2}}\right)^{\frac{1}{k}} \leq n'_k \leq b^{\frac{k}{2}} \left(k!\, k^{\frac{(3+\varepsilon)}{2}}\right)^{\frac{1}{k}} + k + 1. \tag{8}$$

and the asymptotic relation

$$n'_k = b^{\frac{k}{2}} (k!)^{\frac{1}{k}} + o(k b^{\frac{k}{2}}). \tag{9}$$

The asymptotic expressions for n_k and n'_k ensure that $n'_k < n_{k+1}$ when k is sufficiently large.

Proof of (b): It follows from inequalities (6) and (8) that, when k is sufficiently large,

$$\begin{aligned} n'_k - n_k &\leq b^{\frac{k}{2}} \left(k!\, k^{\frac{(3+\varepsilon)}{2}}\right)^{\frac{1}{k}} + k + 1 - \left[b^{\frac{k}{2}} \left(k!\, k^{-\frac{3+\varepsilon}{2}}\right)^{\frac{1}{k}} - 1\right] \\ &\leq b^{\frac{k}{2}} \left(k!\, k^{-\frac{3+\varepsilon}{2}}\right)^{\frac{1}{k}} (k^{\frac{3+\varepsilon}{k}} - 1) + k + 2 \\ &\leq (n_k+1)(k^{\frac{3+\varepsilon}{k}} - 1) + k + 2 \\ &< n_k C_1 \frac{\log k}{k}. \end{aligned}$$

for some constant $C_1 > 2$. The third inequality above is a consequence of (6), while the last inequality follows from the fact that $x - 1 < 2\ln x$ for x close to 1.

Proof of (c): It follows from Equations (7) and (9) that

$$\frac{n_{k+1}}{n_k} = b^{\frac{1}{2}} + o(1) \quad \text{and} \quad \frac{n_{k+2}}{n_{k+1}} = b^{\frac{1}{2}} + o(1).$$

Therefore, as k tends to infinity,

$$\frac{n_{k+2} - n_{k+1}}{n_{k+1} - n_k} = \frac{\frac{n_{k+2}}{n_{k+1}} - 1}{1 - \frac{n_k}{n_{k+1}}} \to b^{\frac{1}{2}}.$$

This completes the proof of Lemma 3. ■

We now continue the analysis of $U_k(n)$. The second moment argument used below requires bounds on the ratio

$$g(U_k(n)) := \mathrm{Var}(U_k(n))/(EU_k(n))^2$$

which arises in a standard Chebyshev bound on the tails of $U_k(n)$. Letting

$$S = \{C = A \times B : A, B \subseteq [n], |A| = |B| = k\}$$

be the family of index sets of $k \times k$ submatrices, we see that

$$U_k(n)^2 \;=\; \sum_{C,C' \in S} I\{\text{each entry of } Z_n[C] \text{ and } Z_n[C'] \text{ is } 1\}.$$

From the last display one may readily derive that

$$EU_k(n)^2 \;=\; \sum_{l=1}^{k} \binom{n}{k}\binom{k}{l}\binom{n-k}{k-l} \sum_{r=1}^{k} \binom{n}{k}\binom{k}{r}\binom{n-k}{k-r} \cdot p^{2k^2-lr},$$

where the indices k and l indicate the number of rows and columns, respectively, that the submatrices C and C' have in common. As $EU_k(n) = \binom{n}{k}^2 p^{k^2}$, we find that

$$g(U_k) \;=\; \sum_{l=0}^{k}\sum_{r=0}^{k} \frac{\binom{k}{l}\binom{n-k}{k-l}}{\binom{n}{k}} \frac{\binom{k}{r}\binom{n-k}{k-r}}{\binom{n}{k}} b^{lr} - 1,$$

where $b = p^{-1}$. Recall that $0 < \varepsilon < \frac{1}{2}$ is fixed.

Lemma 4. *There exists a constant $C_0 > 0$ such that $g(U_k(n)) \le C_0 k^{-1-\varepsilon}$ for every sufficiently large k and every $n_k' \le n \le n_{k+1}$.*

Proof of Lemma 4: To begin, note that

$$\begin{aligned}
g(U_k(n)) &= \sum_{l=0}^{k}\sum_{r=0}^{k} \frac{\binom{k}{l}\binom{n-k}{k-l}}{\binom{n}{k}} \frac{\binom{k}{r}\binom{n-k}{k-r}}{\binom{n}{k}} (b^{lr} - 1) \\
&= \sum_{l=1}^{k}\sum_{r=1}^{k} \frac{\binom{k}{l}\binom{n-k}{k-l}}{\binom{n}{k}} \frac{\binom{k}{r}\binom{n-k}{k-r}}{\binom{n}{k}} (b^{lr} - 1) \\
&< \sum_{l=1}^{k}\sum_{r=1}^{k} \frac{\binom{k}{l}\binom{n-k}{k-l}}{\binom{n}{k}} \frac{\binom{k}{r}\binom{n-k}{k-r}}{\binom{n}{k}} b^{lr} \le \left(\sum_{r=1}^{k} \frac{\binom{k}{r}\binom{n-k}{k-r}}{\binom{n}{k}} (b^{r^2/2}) \right)^2,
\end{aligned}$$

where the last inequality follows from the fact that $b^{lr} \le b^{\frac{l^2+r^2}{2}}$. Thus it suffices to show that

$$\sum_{r=1}^{k} h(r) = O(k^{-1/2-\varepsilon/2}) \;\text{ where }\; h(r) := \frac{\binom{k}{r}\binom{n-k}{k-r}}{\binom{n}{k}} b^{r^2/2}. \tag{10}$$

If $n \ge n_k'$, then by inequality (8), $n \ge b^{\frac{k}{2}} \left(k!\, k^{\frac{3+\varepsilon}{2}} \right)^{\frac{1}{k}}$, which implies that $k \le 2\log_b n$. Similarly, inequality (6) implies that if $n \le n_{k+1}$ then $k \ge (2-\eta)\log_b n$ for some fixed $0 < \eta < 1/2$. Finally,

it follows from the assumption that $n \geq n'_k$ and the definition of n'_k that $\binom{n}{k} p^{\frac{k^2}{2}} = \sqrt{EU_k(n)} \geq \sqrt{EU_k(n'_k)} \geq k^{3/2+\varepsilon/2}$. Using these inequalities, one can upper bound $h(1)$, $h(k-1)$ and $h(k)$ as follows:

$$h(1) \;=\; \frac{\binom{k}{1}\binom{n-k}{k-1}}{\binom{n}{k}} b^{1/2} \;=\; \frac{b^{1/2}k^2(n-k)!(n-k)!}{(n-2k+1)!n!} \;<\; \frac{b^{1/2}k^2}{n-k} = O(k^2 b^{-k/2}),$$

$$h(k-1) \;=\; \frac{k(n-k)}{\binom{n}{k}} b^{\frac{k^2}{2}-k+\frac{1}{2}} \;\leq\; \frac{knb^{\frac{1}{2}-k}}{\sqrt{EU_k(n)}} \;=\; O\left(k^{-1/2-\varepsilon/2} b^{-k(1-\eta)/(2-\eta)}\right),$$

$$h(k) \;=\; \frac{b^{\frac{k^2}{2}}}{\binom{n}{k}} \;=\; \frac{1}{\sqrt{EU_k(n)}} \;\leq\; k^{-3/2-\varepsilon/2},$$

In order to establish inequality (10), it therefore suffices to verify that when k is sufficiently large,

$$h(r) \leq h(1) + h(k-1) \tag{11}$$

for any $1 < r < k-1$. To this end, note that

$$\frac{h(r+1)}{h(r)} = \frac{(k-r)^2 b^{r+\frac{1}{2}}}{(r+1)(n-2k+r+1)}.$$

When $r \leq \frac{1}{3}k$, the inequality $k \leq 2\log_b n$ implies that

$$\frac{h(r+1)}{h(r)} \leq \frac{bk^2 b^{\frac{k}{3}}}{n-2k+r+1} \leq \frac{bk^2 n^{\frac{2}{3}}}{n-2k+r+1} < 1.$$

When $\frac{2}{3}k \leq r < k-1$ the inequality $k \geq (2-\eta)\log_b n$ with $0 < \eta < 1/2$ implies that

$$\frac{h(r+1)}{h(r)} \geq \frac{b^{\frac{2k}{3}}}{k(n+r+1)} \geq \frac{n^{\frac{2(2-\eta)}{3}}}{k(n+r+1)} > 1.$$

In order to show inequality (11), it now suffices to show that $h(r)$ is log-convex for all integer $r \in [\lceil \frac{k}{3} \rceil - 1, \lceil \frac{2k}{3} \rceil]$. Since for $r \in [\lceil \frac{k}{3} \rceil - 1, \lceil \frac{2k}{3} \rceil]$,

$$\ln h(r) = \ln h(\lceil \frac{k}{3} \rceil - 1) + \sum_{i=0}^{r-\lceil \frac{k}{3} \rceil} \ln \frac{h(\lceil \frac{k}{3} \rceil + i)}{h(\lceil \frac{k}{3} \rceil + i - 1)},$$

it is equivalent to show that $\frac{h(r+1)}{h(r)}$ is monotone increasing. To verify this, note that the derivative $\partial[h(r+1)/h(r)]/\partial r$ is equal to

$$\frac{b^{\frac{2r+1}{2}}(k-r)}{(r+1)(n-2k+r+1)} \left[\frac{-2(r+1)(n-2k+r+1) - (k-r)(2r+n-2k+2)}{(r+1)(n-2k+1)} + (k-r)\ln b \right].$$

When k is sufficiently large and $n \gg k > r$, the sum of the leading terms on the last expression above is

$$-2n(r+1) - (k-r)n + (k-r)(r+1)n\ln b \;=\; n(-r^2\ln b + kr\ln b - k - r + (k-r)\ln b - 2).$$

By plugging in $r = \frac{k}{3}$ and $r = \frac{2k}{3}$, it is not hard to check that this quadratic form in r is positive for any $r \in [\lceil \frac{k}{3} \rceil - 1, \lceil \frac{2k}{3} \rceil]$ when k is sufficiently large, and the desired monotonicity follows. ■

Lemma 5. *With probability one, when k is sufficiently large, $M(Z_n) = k$ whenever $n'_k \le n \le n_{k+1}$.*

Proof of Lemma 5: By the definition of n_{k+1} and Markov's inequality, when $n \le n_{k+1}$,

$$P(M(Z_n) > k) \le E(U_{k+1}(n)) \le E(U_{k+1}(n_{k+1})) \le (k+1)^{-3-\varepsilon}.$$

Moreover, Chebyshev's inequality and Lemma 4 together imply that for $n'_k \le n \le n_{k+1}$,

$$P(M(Z_n) < k) = P(U_k(n) = 0) \le \frac{Var(U_k(n))}{(EU_k(n))^2} \le C_0 \cdot k^{-1-\varepsilon}.$$

As $M(Z_n)$ is monotone increasing with n, the previous bounds yield

$$\begin{aligned}\sum_{k\ge 1} P\left(\bigcup_{n=n'_k}^{n_{k+1}} \{M(Z_n) \ne k\}\right) &\le \sum_{k\ge 1} P\left(M(Z_{n'_k}) < k\right) + \sum_{k\ge 1} P\left(M(Z_{n_{k+1}}) \ge k\right) \\ &\le \sum_{k\ge 1}\left(C_0 \cdot k^{-1-\varepsilon} + \frac{1}{k^{3+\varepsilon}}\right) < \infty.\end{aligned}$$

and the result follows from the Borel-Cantelli lemma. ■

Proof of Theorem 1: From Lemma 5 we may deduce that with probability one $M(Z_n)$ is eventually equal to one of two possible consecutive integers, whose values depend only on n. It follows from their definition that $n_k < n'_k$, and by Lemma 3 both integers tend to infinity as k tends to infinity. Therefore for every k greater than or equal to some k_0 we have

$$\ldots < n_k < n'_k < n_{k+1} < n'_{k+1} < \ldots.$$

Thus for all $n \ge n_{k_0}$ there exists a unique integer k (depending on n) such that $n'_k \le n \le n_{k+1}$ or $n_k < n < n'_k$. In the former case, Lemma 5 implies that $M(Z_n) = k$ when n is sufficiently large. In the latter case, Lemma 5 and the monotonicity of $M(Z_n)$ in n imply that

$$k-1 = M(Z_{n_k}) \le M(Z_n) \le M(Z_{n'_k}) = k,$$

when n is sufficiently large, so that $M(Z_n)$ can take one of at most two possible values, $k-1$ and k.

It remains to connect $M(Z_n)$ and $s(n)$. To begin, let n be such that $n'_k \le n \le n_{k+1}$ for some $k \ge k_0$. Then by definition of n_{k+1} and $s(n)$,

$$(1+o(1))\,\phi_n(k+1) = (EU_{k+1}(n))^{1/2} \le (EU_{k+1}(n_{k+1}))^{1/2} \le k^{-3/2-\varepsilon/2} < 1 = \phi_n(s(n)).$$

As $\phi_n(k)$ is monotone decreasing in k, we conclude that $s(n) < k+1$ when n is sufficiently large. Similarly,

$$(1+o(1))\,\phi_n(k) = (E\,U_k(n))^{1/2} \ge (E\,U_k(n'_k))^{1/2} \ge k^{3/2+\varepsilon/2} > 1 = \phi_n(s(n)),$$

which implies $s(n) > k$. Thus, with probability one, when n is sufficiently large

$$n'_k \le n \le n_{k+1} \text{ implies } k < s(n) < k+1 \text{ and } M(Z_n) = k. \tag{12}$$

Suppose now that $n_k \le n \le n'_k$. Then $s(n_k) \le s(n) \le s(n'_k)$ and the arguments above show that $s(n_k) < k$ and $s(n'_k) > k$. We establish that $s(n'_k) - s(n_k) = o(1)$. To this end, note that

$$0 < s(n'_k) - s(n_k) = 2\log_b \frac{n'_k}{n_k} - 2\log_b \frac{\log_b n'_k}{\log_b n_k} + o(1) \le 2\log_b \frac{n'_k}{n_k} + o(1)$$

as $\frac{\log_b n'_k}{\log_b n_k} > 1$. It therefore suffices to show that $\log_b \frac{n'_k}{n_k} = o(1)$, but this follows from part (b) of Lemma 3. Putting the bounds above together with Lemma 5, we find that with probability one, when n is sufficiently large

$$n_k \le n \le n'_k \text{ implies } k - \varepsilon < s(n) < k + \varepsilon \text{ and } M(Z_n) \in \{k-1, k\}. \tag{13}$$

Combining relations (12) and (13) yields the desired bound on $M(Z_n)$. ■

9. Proof of Theorem 4

The following lemmas are used in the proof of Theorem 4. Lemma 6 shows that $|\hat{C}|$ is greater than or equal to $|C^*|$ with high probability, and Lemma 9 shows that $\hat{C}$ can only contain a small proportion of entries outside C^*. Lemma 7 and Lemma 8 are used in the proof of Lemma 9.

Lemma 6. *Under the conditions of Theorem 4,* $P\left(|\hat{C}| < l^2\right) \le \Delta_1(l)$.

Proof of Lemma 6: Let $u_{1*}, \ldots, u_{l*}$ be the rows of C^* in Y, and let V be the number of rows satisfying $F(u_{i*}) < \tau = 1 - p_0$. By the union bound and a standard bound (Devroye et al., 1996) on the tail of the binomial distribution, $P(V \ge 1) \le l \cdot e^{-\frac{3l(p-p_0)^2}{8p}}$. The same inequality holds for the number V' of columns u_{*j} of C^* such that $F(u_{*i}) < 1 - p_0$. Since $\{|\hat{C}| < l^2 = |C^*|\} \subset \{C^* \notin AFI_\tau(Y)\} \subset \{V \ge 1\} \cup \{V' \ge 1\}$, we have

$$\begin{aligned} P\{|\hat{C}| < l^2\} &\le P(V \ge 1) + P(V' \ge 1) \\ &\le 2le^{-\frac{3}{8p}l(p-p_0)^2} = \Delta_1(l). \quad ■ \end{aligned}$$

Lemma 7. *Given* $0 < \tau_0 < 1$, *if there exists a* $k \times r$ *binary matrix* V *such that* $F(V) \ge \tau_0$, *then there exists a* $v \times v$ *submatrix* U *of* V *such that* $F(U) \ge \tau_0$, *where* $v = \min\{k, r\}$.

Proof of Lemma 7: Without loss of generality, assume $v = k \le r$. Order the columns of V in descending order of the number of 1s they contain. If U contains the first v columns in this order, then $F(U) \ge \tau_0$. ■

Lemma 8. *Let* $1 < \gamma < 2$. *Let* W *be a binary matrix, and let* R_1 *and* R_2 *be two square submatrices of* W *such that (i)* $|R_2| = k^2$, *(ii)* $|R_1 \backslash R_2| > k^\gamma$ *and (iii)* $R_1 \in AFI_\tau(W)$. *Then when* k *is sufficiently large there exists a square submatrix* $D \subset R_1 \backslash R_2$ *such that* $|D| \ge k^{2\gamma-2}/16$ *and* $F(D) \ge \tau$.

Proof of Lemma 8: The result is clearly true if $R_1 \cap R_2 = \emptyset$, so we assume that R_1 and R_2 overlap after suitable row and column permutations, $R_1 \backslash R_2$ can be expressed either as a single maximal rectangular submatrix W_1, or as the union of two overlapping maximal rectangular $W_1 \cup W_2$. (A submatrix W of $R_1 \backslash R_2$ is maximal if there is no other submatrix of $R_1 \backslash R_2$ that contains it.)

Case 1: R_1 and R_2 overlap on an edge. Suppose that the difference $R_1 \backslash R_2$ can be expressed as a single rectangular submatrix W_1. Let l_1 and l_2 be the side lengths of W_1. In this case, the side length of the square submatrix R_1 must be less than k, and consequently $\max(l_1, l_2) \leq k$. Since $|R_1 \backslash R_2| \geq k^\gamma$, it follows that $\min(l_1, l_2) \geq k^{\gamma-1}$. As $R_1 \in AFI_\tau(W)$ we have $F(W_1) \geq \tau$. By Lemma 7, there exists a $v \times v$ submatrix D of W_1 such that $F(D) \geq \tau$ and $v \geq \min(l_1, l_2) \geq k^{\gamma-1}$.

Case 2: R_1 and R_2 overlap on a corner. Suppose $R_1 \backslash R_2$ is the union $W_1 \cup W_2$ of two maximal rectangular submatrices. Then clearly $\max(|W_1|, |W_2|) \geq \frac{|R_1 \backslash R_2|}{2}$. Without loss of generality, we assume that $|W_1| \geq |W_2|$. As $R_1 \in AFI_\tau(W)$, $F(W_1) \geq \tau$, and it suffices by Lemma 7 to show that the length of the shorter side of W_1 is greater than $k^{\gamma-1}/4$.

Let $l_1 \leq l_2$ be the side lengths of W_1 and suppose for the moment that $l_1 < k^{\gamma-1}/4$. Then $l_2 > \frac{|R_1 \backslash R_2|}{2k^{\gamma-1}/4}$ and $|R_1| = l_2^2 \geq \frac{|R_1 \backslash R_2|^2}{k^{2\gamma-2}/4}$, and it follows that

$$|R_1 \backslash R_2| \ \geq \ |R_1| - |R_2| \ > \ \frac{|R_1 \backslash R_2|^2}{k^{2\gamma-2}/4} - k^2.$$

Dividing both sides of the previous inequality by $|R_1 \backslash R_2|$ and using the assumption $|R_1 \backslash R_2| \geq k^\gamma$ yields

$$1 \ > \ \frac{|R_1 \backslash R_2|}{k^{2\gamma-2}/4} - \frac{k^2}{|R_1 \backslash R_2|} \ \geq \ 4k^{(2-\gamma)} - k^{(2-\gamma)} \ = \ 3k^{(2-\gamma)}.$$

When k is sufficiently large, this yields a contradiction and completes the proof. ■

Lemma 9. *Let $\mathscr{A}$ be the collection of $C \in \hat{\mathscr{C}}$ such that $|C| \geq l^2$ and $\frac{|C \cap C^{*c}|}{|C|} \geq \alpha$, where $\alpha \in (0,1)$ satisfies $l \geq 8\alpha^{-1}(\log_b n + 2)$. Let A be the event that $\mathscr{A} \neq \emptyset$. If n is sufficiently large,*

$$P(A) \ \leq \ \Delta_2(\alpha, l).$$

Proof of Lemma 9: Recall that $|C^*| = l^2$. If $C \in \mathscr{A}$ then $C \in \mathrm{AFI}_{1-p_0}(Y)$ and

$$|C \backslash C^*| \ = \ |C| \cdot \frac{|C \cap C^{*c}|}{|C|} \ \geq \ l^2 \cdot \alpha \ = \ l^\gamma$$

where $\gamma \ = \ 2 + \log_l \alpha$. Thus, by Lemma 8 there exists a $v \times v$ submatrix D of $C \backslash C^*$ such that $F(D) \ \geq \ 1 - p_0$ and $v \ \geq \ \frac{\alpha l}{4}$. It follows that

$$\max_{c \in \hat{\mathscr{C}}} M^\tau(C \cap C^{*c}) \ \geq \ v \ \geq \ \frac{\alpha l}{4},$$

where $\tau = 1 - p_0$ and $M^\tau(X)$ is size of the largest square submatrix with average greater than τ in a given matrix X.

Let $W = W(Y, C^*)$ be an $n \times n$ binary random matrix, with $w_{ij} = y_{ij}$ if $(i,j) \notin C^*$, and $w_{ij} \sim \mathrm{Bern}(p)$ otherwise. Then it is clear that

$$M^\tau(W) \ \geq \ \max_{c \in \hat{\mathscr{C}}} M^\tau(C \cap C^{*c}) \ \geq \ \frac{\alpha l}{4}.$$

When n is sufficiently large and $l \geq 8\alpha^{-1}(\log_b n + 2)$, we can bound $P(A)$ as follows

$$\begin{aligned} P(A) &\leq P(\max_{c \in \hat{\mathscr{C}}} M^{\tau}(C \cap C^{*c}) \geq \frac{\alpha l}{4}) \\ &\leq P(M^{\tau}(W) \geq \frac{\alpha l}{4}) \leq 2n^{-(\alpha l/4 - 2\log_{b'} n)}, \end{aligned} \tag{14}$$

where $b' = e^{\frac{3(1-p_0-p)^2}{8p}}$. Note that the last inequality follows from a first moment argument similar to that in the proof of Proposition 1 and a standard inequality for the tails of the binomial distribution(cf., Problem 8.3 of Devroye et al. 1996). As $p_0 > p$, $b < b'$, and consequently one can bound the right hand side of inequality (14) by $\Delta_2(\alpha, l)$. For detailed proof of inequality (14), please refer to Proposition 3.3.1 in Sun (2007). ■

Proof of Theorem 4: Let E be the event that $\{\Lambda \leq \frac{1-\alpha}{1+\alpha}\}$. It is clear that E can be expressed as the union of two disjoint events E_1 and E_2, where

$$E_1 = \{|\hat{C}| < |C^*|\} \cap E \text{ and } E_2 = \{|\hat{C}| \geq |C^*|\} \cap E.$$

One can bound $P(E_1)$ by $\Delta_1(l)$ via Lemma 6.

It remains to bound $P(E_2)$. By the definition of Λ, the inequality $\Lambda \leq \frac{1-\alpha}{1+\alpha}$ can be rewritten equivalently as

$$1 + \frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} + \frac{|\hat{C}^c \cap C^*|}{|\hat{C} \cap C^*|} \geq \frac{1+\alpha}{1-\alpha}.$$

When $|\hat{C}| \geq |C^*|$, one can verify that $|\hat{C} \cap C^{*c}| \geq |\hat{C}^c \cap C^*|$, which implies that

$$1 + \frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} + \frac{|\hat{C}^c \cap C^*|}{|\hat{C} \cap C^*|} \leq 1 + 2\frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|}.$$

Therefore, $E_2 \subset E_2^*$, where

$$\begin{aligned} E_2^* &= \{|\hat{C}| \geq |C^*|\} \cap \left\{1 + 2\frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} \geq \frac{1+\alpha}{1-\alpha}\right\} \\ &\subset \{|\hat{C}| \geq l^2\} \cap \left\{1 + 2\frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} \geq \frac{1+\alpha}{1-\alpha}\right\}. \end{aligned}$$

Notice that $1 + 2\frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} \geq \frac{1+\alpha}{1-\alpha}$ implies $\frac{|\hat{C} \cap C^{*c}|}{|\hat{C}|} \geq \alpha$. Therefore, by Lemma 9, $P(E_2^*) \leq \Delta_2(\alpha, l)$. ■

Acknowledgments

The authors would like to thank Professors Gábor Lugosi and Robin Pemantle for helpful discussions regarding early versions of this work, and two referees and the editor for helpful comments and suggestions. Comments from one anonymous referee led to a simpler proof, and improved statement, of Theorem 1. The work presented in this paper was supported in part by NSF grant DMS 0406361.

References

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of data*, pages 207-216, 1993.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. In U. M. Fayyad et. al, editors, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Chapter 12, 307-328, 1996.

R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 94-105, 1998.

N. Alon and J. Spencer. *The Probabilistic Method*. John Wiley. New York, 1991.

B. Bollobás and P. Erdős. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophy Society*, 80:419-427, 1976.

B. Bollobás. *Random Graphs*. 2nd ed., Cambridge University Press, 2001.

Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, pages 93-103, 2000.

M. Dawande, P. Keskinocak, J. Swaminathan, and S. Tayur. On bipartite and multipartite clique problems. *Journal of Algorithms*, 41:388-403, 2001.

L. Devroye, L. Gyorfi, and G, Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1996.

M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-completeness*. Freeman, San Francisco, 1979.

B. Goethals. Survey on Frequent Pattern Mining. `www.adrem.ua.ac.be/~goethals/software/survey.pdf.2003`.

J. Han, J. Pei and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 1-12, 2000.

D. J. Hand, H. Mannila and P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

D. S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29(1):174-200, 1998.

M. Koyutürk, W. Szpankowski and A. Grama. Biclustering gene-feature matrices for statistically significant dense patterns. In *Proceedings of the 8th Annual International Conference on Research in Computational Molecular Biology*, pages 480-484, 2004.

J. Liu, S. Paulsen, W. Wang, A. B. Nobel, and J. Prins. Mining approximate frequent itemsets from noisy data. In *Proceedings of the IEEE International Conference on Data Mining*, pages 721-724, 2005.

J. Liu, S. Paulsen, X. Sun, W. Wang, A.B. Nobel, and J. Prins. Mining approximate frequent itemsets in the presence of noise: algorithm and analysis. In *Proceedings of the SIAM International Conference on Data Mining*, pages 405-416, 2006.

S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):24-45, 2004.

D. Matula. The largest clique size in a random graph. CS 7608, Technical Report, Southern Methodist University, 1976.

N. Mishra, D. Ron, and R. Swaminathan. A new conceptual clustering framework. *Machine Learning*. 56(1-3):115-151, 2004.

G. Park and W. Szpankowski. Analysis of biclusters with applications to gene expression data. In *Proceedings of Conference on Analysis of Algorithms*, CS 7608, pages 267-274, 2005.

R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651-654, 2003.

J. Pei, A. K. Tung, and J. Han. Fault-tolerant frequent pattern mining: Problems and challenges. In *Proceedings of the ACM SIGMOD International Workshop on Research Issues on Data Mining and Knowledge Disco*, 2001.

J. Pei, G. Dong, W. Zou, and J. Han. Mining condensed frequent-pattern bases. *Knowledge and Information Systems*, 6(5):570-594, 2002.

J. D. Reuning-Scherer. *Mixture Models for Block Clustering.* Ph.D. Thesis, Yale university, 1997.

J. K. Seppänen, and H. Mannila. Dense itemsets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 683-688, 2004.

X. Sun. *Significance and Recovery of Block Structures in Binary and Real-valued Matrices with Noise.* Ph.D. Thesis, UNC Chapel Hill, 2007.

A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18:136-144, 2002

A. Tanay, R. Sharan and R. Shamir. Biclustering algorithms: A survey. *Handbook of Computational Molecular Biology*, Chapman & Hall/CRC, Computer and Information Science Series, 2005.

C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 194-203, 2001.

Journal of Machine Learning Research 9 (2008) 2455-2487 Submitted 9/07; Revised 5/08; Published 11/08

Minimal Nonlinear Distortion Principle for Nonlinear Independent Component Analysis

Kun Zhang KZHANG@CSE.CUHK.EDU.HK
Laiwan Chan LWCHAN@CSE.CUHK.EDU.HK
Department of Computer Science and Engineering
The Chinese University of Hongkong
Hong Kong

Editor: Aapo Hyvärinen

Abstract

It is well known that solutions to the nonlinear independent component analysis (ICA) problem are highly non-unique. In this paper we propose the "minimal nonlinear distortion" (MND) principle for tackling the ill-posedness of nonlinear ICA problems. MND prefers the nonlinear ICA solution with the estimated mixing procedure as close as possible to linear, among all possible solutions. It also helps to avoid local optima in the solutions. To achieve MND, we exploit a regularization term to minimize the mean square error between the nonlinear mixing mapping and the best-fitting linear one. The effect of MND on the inherent trivial and non-trivial indeterminacies in nonlinear ICA solutions is investigated. Moreover, we show that local MND is closely related to the smoothness regularizer penalizing large curvature, which provides another useful regularization condition for nonlinear ICA. Experiments on synthetic data show the usefulness of the MND principle for separating various nonlinear mixtures. Finally, as an application, we use nonlinear ICA with MND to separate daily returns of a set of stocks in Hong Kong, and the linear causal relations among them are successfully discovered. The resulting causal relations give some interesting insights into the stock market. Such a result can not be achieved by linear ICA. Simulation studies also verify that when doing causality discovery, sometimes one should not ignore the nonlinear distortion in the data generation procedure, even if it is weak.

Keywords: nonlinear ICA, regularization, minimal nonlinear distortion, mean square error, best linear reconstruction

1. Introduction

Independent component analysis (ICA) is a popular statistical technique aiming to recover independent sources from their observed mixtures, without knowing the mixing procedure or any specific knowledge of the sources (Hyvärinen et al., 2001; Cardoso, 1998; Cichocki and Amari, 2003). In the case that the observed mixtures are a linear transformation of the sources, under weak assumptions, ICA can recover the original sources with the trivial permutation and scaling indeterminacies. Linear ICA is currently a popular method for blind source separation (BSS) of linear mixtures.

However, nonlinear ICA does not necessarily lead to nonlinear BSS. In Hyvärinen and Pajunen (1999), it was shown that solutions to nonlinear ICA always exist and that they are highly non-unique. Actually, one can easily construct a nonlinear transformation of some non-Gaussian independent variables to produce independent outputs. Below are a few examples. Let $y_1, ..., y_n$ be some independent variables. Their component-wise nonlinear functions are still mutually indepen-

dent. If we use Gaussianization (Chen and Gopinath, 2001) to transform y_i into Gaussian variables u_i, any component-wise nonlinear function of $\mathbf{U} \cdot \mathbf{u}$, where $\mathbf{u} = (u_1, ..., u_n)^T$ and $\mathbf{U}$ is an orthogonal matrix, still has mutually independent components. Taleb and Jutten (1999) also gave an example in which nonlinear mixtures of independent variables are still independent. One can see that nonlinear BSS is impossible without additional prior knowledge on the mixing model, since the independence assumption is not strong enough in the general nonlinear mixing case (Jutten and Taleb, 2000; Taleb, 2002).

If we constrain the nonlinear mixing mapping to have some specific forms, the indeterminacies in the results of nonlinear ICA can be reduced dramatically, and as a consequence, nonlinear ICA may lead to nonlinear BSS. For example, in Burel (1992), a parametric form of the mixing transformation is assumed known and one just needs to adjust the unknown parameters. The learning algorithms were improved in Yang et al. (1998). By exploiting the extensions of the Darmois-Skitovich theorem (Kagan et al., 1973) to nonlinear functions, a particular class of nonlinear mixing mappings, which satisfy an addition theorem in the sense of the theory of functional equations, were considered in Eriksson and Koivunen (2002). In particular, the post-nonlinear (PNL) mixing model (Taleb and Jutten, 1999), which assumes that the mixing mapping is a linear transformation followed by a component-wise nonlinear one, has drawn much attention.

In practice, the exact form of the nonlinear mixing procedure is probably unknown. Consequently, in order to model arbitrary nonlinear mappings, one may need to resort to a flexible nonlinear function approximator, such as the multi-layer perceptron (MLP) or the radial basis function (RBF) network, to represent the nonlinear separation system. Almeida (2003) uses the MLP to model the separation system and trains the MLP by information-maximization (Infomax). Moreover, the smoothness constraint,[1] which is implicitly provided by MLP's with small initial weights and with a relatively small number of hidden units, was believed to be a suitable regularization condition to achieve nonlinear BSS. In Tan et al. (2001), a RBF network is adopted to represent the separation system, and partial moments of the outputs of the separation system are used for regularization. The matching between the relevant moments of the outputs and those of the original sources was expected to guarantee a unique solution. But the moments of the original sources may be unknown. In addition, if the transformation from the original sources to the recovered sources is non-trivial,[2] this regularization could not help to recover the original sources. Variational Bayesian nonlinear ICA (Lappalainen and Honkela, 2000; Valpola, 2000) uses the MLP to model the nonlinear mixing transformation. By resorting to the variational Bayesian inference technique, this method can do model selection and avoid overfitting. If we can have some additional knowledge about the nonlinear mixing transformation and incorporate it efficiently, the results of nonlinear ICA will be much more meaningful and reliable.

Although we may not know the form of the nonlinearity in the data generation procedure, fortunately, in many cases the nonlinearity for generating natural signals we deal with is not strong. Hence, provided that the nonlinear ICA outputs are mutually independent, we would prefer the solution with the estimated data generation procedure of minimal nonlinear distortion (MND). This

1. Following Tikhonov and Arsenin (1977), here we use the term "smoothness" in a very general sense. Often it means that that the function does not change abruptly and/or that it does not oscillate too much.
2. For the definition of a trivial transformation, one may see Jutten and Taleb (2000). A one-to-one mapping $\mathcal{H}$ is trivial if and only if it satisfies $\mathcal{H}_i(y_1, y_2, ..., y_n) = h_i(y_{\sigma(i)})$, $i = 1, 2, ..., n$, where h_i are arbitrary functions and σ is any permutation over $\{1, .2, ..., n\}$. That is, a trivial mapping of $\mathbf{y}$ is a permutation of y_i followed by a component-wise transformation.

information can help to reduce the indeterminacies in nonlinear ICA greatly, and moreover, to avoid local optima in the solutions to nonlinear ICA. The minimal nonlinear distortion of the mixing system is achieved by the technique of regularization. The objective function of nonlinear ICA with MND is the mutual information between outputs penalized by some terms measuring the level of "closeness to linear" of the mixing system. The mean square error (MSE) between the nonlinear mixing system and its best-fitting linear one provides such a regularization term. To ensure that nonlinear ICA results in nonlinear BSS, one may also need to enforce the local MND of the nonlinear mapping averaged at every point, which turns out to be the smoothness regularizer exploiting second-order partial derivatives.

MND, as well as the smoothness regularizer, can be incorporated in various nonlinear ICA methods to improve the results. Here we consider two nonlinear ICA methods. The first one is the MISEP method (Almeida, 2003), where the MLP is used to represent the separation system. As regularization is powerful for complexity control in neural networks (Bishop, 1995), the structure of the MLP is not optimized during the learning process, that is, it is fixed. The second one is nonlinear ICA based on kernels (Zhang and Chan, 2007a), in which the nonlinear separation system is modeled using some kernel methods. We then explain why MND helps to alleviate the ill-posedness in nonlinear ICA solutions, by investigating the effect of MND on trivial and non-trivial indeterminacies in nonlinear ICA solutions systematically. Next, we conduct experiments using synthetic data to compare the performance of several nonlinear ICA methods. The results confirm the effectiveness of the proposed MND principle to avoid unwanted solutions and to improve the separation performance. Finally, nonlinear ICA with MND is used to discover linear causal relations in the Hong Kong stock market and give encouraging results. We also give experimental results on synthetic data, which illustrate that when performing ICA-based causality discovery on the data whose generation procedure involves nonlinear distortion, one should take into account the nonlinear effect in the ICA separation system, even if it is mild.[3]

2. Nonlinear ICA with Minimal Nonlinear Distortion

In this section we first briefly review the general nonlinear ICA problem, and then propose the minimal nonlinear distortion (MND) principle for regularization of nonlinear ICA.

2.1 Nonlinear ICA

In the nonlinear ICA model, the observed data $\mathbf{x} = (x_1, ..., x_n)^T$ are assumed to be generated from a vector of independent variables $\mathbf{s} = (s_1, ..., s_n)^T$ by a nonlinear transformation:

$$\mathbf{x} = \mathcal{F}(\mathbf{s}), \tag{1}$$

where $\mathcal{F}$ is an unknown real-valued n-component mixing function. Here for simplicity, we have assumed that the number of observed variables equals that of the original independent variables. The general nonlinear ICA problem is to find a mapping $\mathcal{G} : \mathbb{R}^n \to \mathbb{R}^n$ such that

$$\mathbf{y} = \mathcal{G}(\mathbf{x})$$

has statistically independent components. As mentioned in Section 1, the results of nonlinear ICA are highly non-unique. In order to achieve nonlinear BSS, which aims at recovering the original sources s_i, we should resort to additional prior information or suitable regularization constraints.

3. Some preliminary results of this paper were presented at ICML2007 (Zhang and Chan, 2007b).

2.2 With Minimum Nonlinear Distortion

We now propose the MND principle to restrict the space of nonlinear ICA solutions. As a consequence, the ill-posedness of the nonlinear ICA problem is alleviated. Under the condition that the separation outputs y_i are mutually independent, this principle prefers the solution with the estimated mixing transformation $\hat{\mathcal{F}}$ as close as possible to linear.

Now we need a measure of "closeness to linear" of a mapping. Given a nonlinear mapping $\hat{\mathcal{F}}$, its deviation from the affine mapping $\mathbf{A}^*$, which fits $\hat{\mathcal{F}}$ best among all affine mappings $\mathbf{A}$, is an indicator of its "closeness to linear", or the level of its nonlinear distortion. The deviation can be measured in various ways. The MSE is adopted here, as it greatly facilitates subsequent analysis. Consequently, the "closeness to linear" of $\hat{\mathcal{F}} = \mathcal{G}^{-1}$ can be measured by the MSE between $\mathcal{G}^{-1}$ and $\mathbf{A}^*$. We denote this measure by $R_{MSE}(\theta)$, where θ denotes the set of unknown parameters in the nonlinear ICA system. Let $\mathbf{x}^* = (x_1^*, \cdots, x_n^*)^T$ be the output of the affine transformation from $\mathbf{y}$ by $\mathbf{A}^*$. Let $\tilde{\mathbf{y}} = [\mathbf{y}; 1]$. $R_{MSE}(\theta)$ can then be written as the MSE between x_i and x_i^*:

$$\begin{aligned} R_{MSE}(\theta) &= E\{(\mathbf{x}-\mathbf{x}^*)^T(\mathbf{x}-\mathbf{x}^*)\}\text{, where} \\ \mathbf{x}^* &= \mathbf{A}^*\tilde{\mathbf{y}}, \text{ and } \mathbf{A}^* = \arg_{\mathbf{A}} \min E\{(\mathbf{x}-\mathbf{A}\mathbf{y})^T(\mathbf{x}-\mathbf{A}\mathbf{y})\}. \end{aligned} \tag{2}$$

Here $\mathbf{A}^*$ is a $n \times (n+1)$ matrix.[4] Figure 1 shows the separation system $\mathcal{G}$ together with the generation process of R_{MSE}.

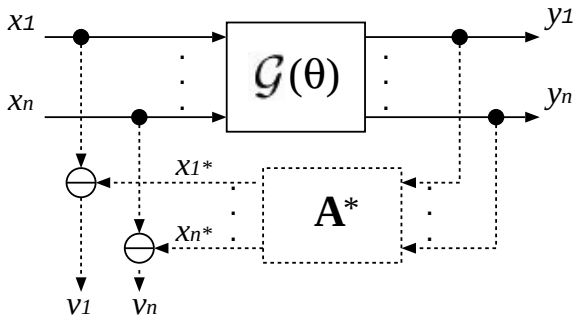


Figure 1: The separation system $\mathcal{G}$ (solid line) and the generation of the regularization term R_{MSE} (dashed line). $R_{MSE} = \sum_{i=1}^{n} v_i^2$, where $v_i = x_i - x_i^*$.

With R_{MSE} measuring the level of nonlinear distortion, nonlinear ICA with MND can be formulated as the following constrained optimization problem. It aims to minimize the mutual information between outputs, that is, $I(\mathbf{y})$, subject to $R_{MSE}(\theta) \le t$, where t is a pre-assigned parameter. The Lagrangian for this optimization problem is $L(\theta, \lambda) = I(\mathbf{y}) + \lambda[R_{MSE}(\theta) - t]$ with $\lambda \ge 0$. To find θ, we need to minimize

$$J = I(\mathbf{y}) + \lambda R_{MSE}(\theta). \tag{3}$$

The non-negative constant λ depends on the pre-assigned parameter t.

Another advantage of the MND principle is that it tends to make the mapping $\mathcal{G}$ invertible. In the general nonlinear ICA problem, it is assumed that both $\mathcal{F}$ and $\mathcal{G}$ are invertible. But in practice

4. If $E(\mathbf{y}) = E(\mathbf{x}) = \mathbf{0}$, $\mathbf{x}^*$ can be obtained as $\mathbf{x}^* = \mathbf{A}^*\mathbf{y}$ instead, and here $\mathbf{A}^*$ is a $n \times n$ matrix.

it is not easy to guarantee the invertibility of the mapping provided by a flexible nonlinear function approximator, like the MLP. MND pushes $\mathcal{G}$ to be close to a linear invertible transformation. Hence when nonlinearity in $\mathcal{F}$ is not too strong, MND helps to guarantee the invertibility of the nonlinear ICA separation system $\mathcal{G}$.

2.2.1 SIMPLIFICATION OF R_{MSE}

R_{MSE}, given in Eq. 2, can be further simplified. According to Eq. 2, the derivative of R_{MSE} w.r.t. $\mathbf{A}^*$ is $\frac{\partial R_{MSE}}{\partial \mathbf{A}^*} = -2E\{(\mathbf{x}-\mathbf{A}^*\tilde{\mathbf{y}})\tilde{\mathbf{y}}^T\}$. Setting the derivative to $\mathbf{0}$ gives $E\{(\mathbf{x}-\mathbf{A}^*\tilde{\mathbf{y}})\tilde{\mathbf{y}}^T\} = \mathbf{0}$, which implies

$$\mathbf{A}^* = E\{\mathbf{x}\tilde{\mathbf{y}}^T\}[E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\}]^{-1}. \tag{4}$$

We can see that due to the adoption of the MSE, $\mathbf{A}^*$ can be obtained in closed form. This greatly simplifies the derivation of learning rules.

Due to Eq. 4, we have $E\{\mathbf{A}^*\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\mathbf{A}^{*T}\} = E\{\mathbf{x}\tilde{\mathbf{y}}^T\}\mathbf{A}^{*T}$,and R_{MSE} then becomes

$$\begin{aligned} R_{MSE} &= \mathrm{Tr}\big(E\{(\mathbf{x}-\mathbf{A}^*\tilde{\mathbf{y}})(\mathbf{x}-\mathbf{A}^*\tilde{\mathbf{y}})^T\}\big) \\ &= \mathrm{Tr}\big(E\{\mathbf{x}\mathbf{x}^T - \mathbf{A}^*\tilde{\mathbf{y}}\mathbf{x}^T - \mathbf{x}\tilde{\mathbf{y}}^T\mathbf{A}^{*T} + \mathbf{A}^*\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\mathbf{A}^{*T}\}\big) \\ &= \mathrm{Tr}\big(E\{\mathbf{x}\mathbf{x}^T - \mathbf{A}^*\tilde{\mathbf{y}}\mathbf{x}^T - \mathbf{x}\tilde{\mathbf{y}}^T\mathbf{A}^{*T} + \mathbf{x}\tilde{\mathbf{y}}^T\mathbf{A}^{*T}\}\big) \\ &= -\mathrm{Tr}\big(E\{\mathbf{A}^*\tilde{\mathbf{y}}\mathbf{x}^T\}\big) + \mathrm{Tr}\big(E\{\mathbf{x}\mathbf{x}^T\}\big) \\ &= -\mathrm{Tr}\big(E\{\mathbf{x}\tilde{\mathbf{y}}^T\}[E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\}]^{-1}E\{\tilde{\mathbf{y}}\mathbf{x}^T\}\big) + \text{const.} \end{aligned} \tag{5}$$

Since y_i are independent from each other, they are uncorrelated. We can also easily make y_i zero-mean. Consequently, $E\{\tilde{\mathbf{y}}\tilde{\mathbf{y}}^T\} = \mathrm{diag}\{E(y_1^2), E(y_2^2), ..., E(y_n^2), 1\}$, and R_{MSE} becomes

$$\begin{aligned} R_{MSE} &= -\mathrm{Tr}\big(E\{\mathbf{x}\tilde{\mathbf{y}}^T\} \cdot [\mathrm{diag}\{E(y_1^2), ..., E(y_n^2), 1\}]^{-1} \cdot E\{\tilde{\mathbf{y}}\mathbf{x}^T\}\big) + \text{const} \\ &= -\sum_{j=1}^{n}\sum_{i=1}^{n} \frac{E^2(x_j y_i)}{E(y_i^2)} + \text{const.} \end{aligned} \tag{6}$$

R_{MSE} depends only on the inputs and the outputs of the nonlinear ICA system $\mathcal{G}(\theta)$. Given a form for $\mathcal{G}$, the learning rule for nonlinear ICA with MND is derived by minimizing Eq. 3. Note that R_{MSE}, given in Eq. 2, is inconsistent with certain scaling properties of the observations $\mathbf{x}$. To avoid this, one needs to normalize the variance of the observations x_i through preprocessing, if necessary.

2.2.2 DETERMINATION OF THE REGULARIZATION PARAMETER λ

We suggest initializing λ with a large value λ_0 at the beginning of training and decreasing it to a small constant λ_c during the learning process. A large value for λ at the beginning reduces the possibility of getting into unwanted solutions, which may be non-trivial transformations of the original sources s_i or local optima. As training goes on, the influence of the regularization term is relaxed, and $\mathcal{G}$ gains more freedom. Hopefully, nonlinearity will be introduced, if necessary. The choice of λ_c depends on the level of nonlinear distortion in the mixing procedure. If the nonlinear distortion is considerable, we should use a very small value for λ_c to give the $\mathcal{G}$ network enough flexibility. In our experiments, we found that the separation performance of nonlinear ICA with MND is robust to the value of λ_c in a certain range. If the variance of the observations x_i is normalized, typical values used in our experiments are $\lambda_0 = 5$ and $\lambda_c = 0.01$.

2.3 Relation to Previous Works

The MISEP method has been reported to solve some nonlinear BSS problems successfully, including separating a real-life nonlinear image mixture (Almeida, 2005, 2003). Almeida (2003) claimed that the MLP itself may provide suitable regularization for nonlinear ICA. Some means were also used for regularization in the experiments there. For example, first, direct connections between inputs and output units were incorporated in the $\mathcal{G}$ network. Direct connections can quickly adapt the linear part of the mapping $\mathcal{G}$. Second, in Almeida (2005), the $\mathcal{G}$ network was initialized with an identity mapping, and during the first 100 epochs, it was constrained to be linear (by keeping the output weights of the hidden layer equal to zero). After that, the $\mathcal{G}$ network began learning the nonlinear distortion. $\mathcal{G}$ is therefore expected to be not far from linear, and MND is achieved to some extent. Accordingly, nice experimental results reported there could support the usefulness of the MND principle. We should mention that the MND principle formulated here, as well as the corresponding regularizer, provides a way to control the nonlinearity of the mixing mapping. It can be incorporated by any nonlinear ICA method, including MISEP. Later, we will investigate the effect of MND on nonlinear ICA solutions theoretically, and compare various related nonlinear ICA methods empirically.

In the kernel-based nonlinear BSS method (Harmeling et al., 2003), the data are first mapped to a high-dimensional kernel feature space. Next, a BSS method based on second order temporal decorrelation is performed. In this way a large number of components are extracted. When the nonlinearity in data generation is not too strong, the MND principle provides a way to select a subset of output components corresponding to the original sources. Assume that the outputs y_i are made zero-mean and of unit variance. From Eq. 6 we can see that one can select y_i with large $\sum_{j=1}^{n} \frac{E^2(x_j y_i)}{E(y_i^2)} = \sum_{j=1}^{n} E^2(x_j y_i) = \sum_{j=1}^{n} \mathrm{var}(x_j) \cdot \mathrm{corr}^2(x_j, y_i)$.

It is worth mention that the principle of least mean square error reconstruction has been used for training a class of neural networks and gives some interesting results (Xu, 1993). For one-layer networks with linear/nonlinear units, this principle leads to principal component analysis (PCA)/ICA. We should address that the reconstruction in their work is quite different from that discussed in Section 2.2 in this paper. In their work, the forward process and the reconstruction process share the same weights; in this paper, reconstructed signals are an affine mapping of the outputs, and parameters in the affine mapping are determined by minimizing the reconstruction error.

Smoothness provides a constraint to prevent a neural network from overfitting noisy data. It is also useful to ensure nonlinear ICA to result in nonlinear BSS (Almeida, 2003). In fact, the smoothness regularizer exploiting second-order derivatives (Tikhonov and Arsenin, 1977; Poggio et al., 1985) is also related to the MND principle, as shown below.

2.4 Local Minimal Nonlinear Distortion: Smoothness

R_{MSE}, given in Eq. 2, indicates the deviation of the mapping $\hat{\mathcal{F}}$ from the affine mapping which fits $\hat{\mathcal{F}}$ *globally* best. In contrast, one may enforce the *local* MND of the nonlinear mapping averaged at every point. We will show that this regularization actually leads to the smoothness regularizer exploiting second-order partial derivatives (Tikhonov and Arsenin, 1977; Poggio et al., 1985; Bishop, 1993).

For a one-dimensional sufficiently smooth function $g(\mathbf{x})$, we can use the second-order Taylor expansion to approximate its function value in the vicinity of $\mathbf{x}$ in terms of $g(\mathbf{x})$:

$$g(\mathbf{x}+\varepsilon) \approx g(\mathbf{x}) + \Big(\frac{\partial g}{\partial \mathbf{x}}\Big)^T \cdot \varepsilon + \frac{1}{2}\varepsilon^T \mathbf{H}_{\mathbf{x}} \varepsilon,$$

where ε is a small variation of $\mathbf{x}$ and $\mathbf{H}_{\mathbf{x}}$ denotes the Hessian matrix of g. Let $\bigtriangledown_{ij} = \frac{\partial^2 g}{\partial x_i \partial x_j}$. If we use the first-order Taylor expansion of g, which is a linear function, to approximate $g(\mathbf{x}+\varepsilon)$, the square error is

$$\begin{aligned} & \Big|\Big| g(\mathbf{x}+\varepsilon) - g(\mathbf{x}) - \Big(\frac{\partial g}{\partial \mathbf{x}}\Big)^T \cdot \varepsilon \Big|\Big|^2 \approx \frac{1}{4}\big|\big|\varepsilon^T \mathbf{H}_{\mathbf{x}} \varepsilon\big|\big|^2 = \frac{1}{4}\Big(\sum_{i,j=1}^{n} \bigtriangledown_{ij} \varepsilon_i \varepsilon_j\Big)^2 \\ \leq \; & \frac{1}{4}\Big(\sum_{i,j=1}^{n} \bigtriangledown_{ij}^2\Big)\Big(\sum_{i,j=1}^{n} \varepsilon_i^2 \varepsilon_j^2\Big) = \frac{1}{4}\Big(\sum_{i,j=1}^{n} \bigtriangledown_{ij}^2\Big)\Big(\sum_{i=1}^{n} \varepsilon_i^2\Big)^2 = \frac{1}{4}||\varepsilon||^4 \cdot \sum_{i,j=1}^{n} \bigtriangledown_{ij}^2. \end{aligned}$$

The above inequality holds due to the Cauchy's inequality. Now we can see that in order to achieve the local MND of g averaged in the domain of $\mathbf{x}$, we just need to minimize the following

$$\int_{\mathbb{D}_{\mathbf{x}}} \sum_{i,j=1}^{n} \bigtriangledown_{ij}^2 d\mathbf{x} = \int_{\mathbb{D}_{\mathbf{x}}} \Big(\sum_{i=1}^{n} \bigtriangledown_{ii}^2 + 2 \sum_{\substack{i,j=1,\\ i<j}}^{n} \bigtriangledown_{ij}^2\Big) d\mathbf{x}. \tag{7}$$

This regularizer has been used for achieving the smoothness constraint (see, e.g., Grimson 1982 for its application in computer vision). When the mapping is vector-valued, we need to apply the above regularizer to each component of the mapping.

Originally we intended to do regularization on the mixing mapping $\hat{\mathcal{F}}$, but it is difficult to do since it is hard to evaluate $\frac{\partial^2 x_l}{\partial y_i \partial y_j}$. Instead, we do regularization on $\mathcal{G}$, the inverse of $\hat{\mathcal{F}}$. The regularization term in Eq. 3 then becomes

$$R_{local}(\theta) = \int_{\mathbb{D}_{\mathbf{x}}} \sum_{l=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \Big(\frac{\partial^2 y_l}{\partial x_i \partial x_j}\Big)^2 d\mathbf{x} = \int_{\mathbb{D}_{\mathbf{x}}} \sum_{i=1}^{n} \sum_{j=1}^{n} P_{ij} d\mathbf{x}, \tag{8}$$

where $P_{ij} \triangleq \sum_{l=1}^{n} \Big(\frac{\partial^2 y_l}{\partial x_i \partial x_j}\Big)^2$. Nonlinear ICA with a smooth de-mixing mapping can be achieved by minimizing the mutual information between y_i, with R_{local}, given by Eq. 8, as the regularization term. There are totally $\frac{n^2(n+1)}{2}$ different terms $\Big(\frac{\partial^2 y_l}{\partial x_i \partial x_j}\Big)^2$ in the integrand of R_{local}. For simplicity and computational reasons, sometimes one may drop the cross derivatives in Eq. 8, that is, $\Big(\frac{\partial^2 y_l}{\partial x_i \partial x_j}\Big)^2$ with $i \neq j$, and consequently obtain the curvature-driven smoothing regularizer proposed in Bishop (1993), with the number of different terms in the integrand being n^2.

3. Incorporation of MND in Different Nonlinear ICA Methods

Now we should choose a model for the nonlinear ICA separation system $\mathcal{G}(\theta)$ and give the learning rule for nonlinear ICA with MND as well as nonlinear ICA with the smoothness constraint for $\mathcal{G}$. Two nonlinear ICA methods are considered here. They are MISEP (Almeida, 2003) and nonlinear ICA based on kernels (Zhang and Chan, 2007a).

3.1 MISEP with MND

Before incorporating MND into the MISEP method (Almeida, 2003) for nonlinear ICA, we give an overview of this method.

3.1.1 MISEP FOR NONLINEAR ICA

MISEP adopts the MLP to model the separation function $\mathcal{G}$ in the nonlinear ICA problem. Figure 2 shows the structure used in this method. This method extends the original Infomax method for linear ICA (Bell and Sejnowski, 1995) in two aspects. First, the separation system is a nonlinear transformation, which is modeled by the MLP. Second, the nonlinearities ψ_i are not fixed in advance, but tuned by the Infomax principle, together with $\mathcal{G}$.

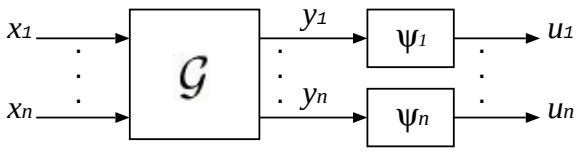


Figure 2: The network structure used in Infomax and MISEP. $\mathcal{G}$ is the separation system, and ψ_i are the nonlinearities applied to the separated signals. In MISEP, $\mathcal{G}$ is a nonlinear transformation, and both $\mathcal{G}$ and ψ_i are learned by the Infomax principle.

With the Infomax principle, parameters in $\mathcal{G}$ and ψ_i are learned by maximizing the joint entropy of the outputs of the structure in Figure 2, which can be written as $H(\mathbf{u}) = H(\mathbf{x}) + E\{\log|\det \mathbf{J}|\}$, where $\mathbf{J} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ is the Jacobian of the nonlinear transformation from $\mathbf{x}$ to $\mathbf{u}$. As $H(\mathbf{x})$ does not depend on the parameters in $\mathcal{G}$ and ψ_i, it can be considered as a constant. Maximizing $H(\mathbf{u})$ is thus equivalent to minimizing

$$J_1(\theta) = -E\{\log|\det \mathbf{J}|\}, \tag{9}$$

where θ denotes the set of unknown parameters. The learning rules for θ were derived by Almeida (2003), in a manner similar to the back-propagation algorithm.

The MLP adopted in this paper has linear output units and a single hidden layer. For the hidden units, the activation function $l(\cdot)$ may be the logistic sigmoid function, the arctan function, etc. Direct connections between the inputs and output units are also allowed. Let $\mathbf{a} = [a_1, ..., a_M]^T$ be the inputs to the hidden units, $\mathbf{z} = [z_1, ..., z_M]^T$ be the output of the hidden units, and $\mathbf{W}$ and $\mathbf{b}$ denote the weights and biases, respectively. We use superscripts to distinguish the locations of these parameters: $\mathbf{W}^{(d)}$ denotes the weights from the inputs to output units, $\mathbf{W}^{(1)}$ those from the inputs to the hidden layer, and $\mathbf{W}^{(2)}$ those from the hidden layer to the output units. $\mathbf{b}^{(1)}$ and $\mathbf{b}^{(2)}$ are the bias vectors in the hidden layer and in the output units, respectively. The output of the $\mathcal{G}$ network represented by this MLP takes the form:

$$\begin{aligned} \mathbf{y} &= \mathbf{W}^{(2)} \cdot \mathbf{z} + \mathbf{W}^{(d)}\mathbf{x} + \mathbf{b}^{(2)}, \text{ where} \\ z_i &= l(a_i), \text{ and } \mathbf{a} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}. \end{aligned} \tag{10}$$

3.1.2 MISEP WITH MND

For MISEP with MND, the objective function to be minimized is Eq. 9 regularized by R_{MSE} given in Eq. 6. The learning rule for θ to minimize Eq. 9 has been considered in Almeida (2003). Hence here we only give the gradient of R_{MSE} w.r.t. θ.

Using the chain rule, also noting Eq. 10, the gradient of $R_{MSE}(\theta)$ w.r.t. $\mathbf{W}^{(2)}$ can be obtained:

$$\frac{\partial R_{MSE}}{\partial \mathbf{W}^{(2)}} = E\Big\{\sum_{i=1}^{n} 2\Big[\frac{E^2(x_j y_i)}{E^2(y_i^2)} y_i - \frac{E(x_j y_i)}{E(y_i^2)} x_i\Big] \cdot \frac{\partial y_i}{\partial \mathbf{W}^{(2)}}\Big\} = E\{\mathbf{K} \cdot \mathbf{z}^T\}, \tag{11}$$

where $\mathbf{K} \triangleq [K_1, ..., K_n]^T$ with its i-th element being $K_i = 2\sum_j \left[\frac{E^2(x_j y_i)}{E^2(y_i^2)} y_i - \frac{E(x_j y_i)}{E(y_i^2)} x_j\right]$, and $\mathbf{z} = [z_1, z_2, ..., z_M]^T$ is the output of the hidden layer of the MLP. For the gradient of R_{MSE} w.r.t. $\mathbf{W}^{(1)}$, $\mathbf{W}^{(d)}$, $\mathbf{b}^{(2)}$, and $\mathbf{b}^{(1)}$, see Appendix A.

3.1.3 MISEP WITH SMOOTHNESS CONSTRAINT ON $\mathcal{G}$

The mapping provided by a MLP may not be smooth enough to make nonlinear ICA result in nonlinear BSS. So here we also implement MISEP with the smoothness constraint on $\mathcal{G}$. The objective function to be minimized becomes Eq. 9 regularized by R_{local} given in Eq. 8. P_{ij} appears in the expression of R_{local}. We first derive its gradient w.r.t. θ in a way analogous to that in Bishop (1993); see Appendix B.

In calculation of $\frac{\partial R_{local}}{\partial \theta}$, the integral in Eq. 8 is difficult to evaluate. Below are two ways to tackle this problem. A very simple way to approximate Eq. 7 is to use the average of the integrand over all observations instead of the integral (ignoring a constant scaling factor), just as Bishop (1993) does:

$$R_{local}^{(1)}(\theta) = E\Big\{\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}\Big\}. \tag{12}$$

This approximation actually assumes that the distribution of $\mathbf{x}$ is close to uniform, as seen from below. Eq. 8 can be rewritten as

$$R_{local}(\theta) = \int_{\mathbb{D}_\mathbf{x}} p(\mathbf{x}) \cdot \frac{1}{p(\mathbf{x})} \sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij} d\mathbf{x} = E\Big\{\frac{1}{p(\mathbf{x})}\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}\Big\}. \tag{13}$$

If $p(\mathbf{x})$ is a constant in the domain $\mathbb{D}_\mathbf{x}$, Eq. 12 is equivalent to Eq. 13; otherwise, the approximation using Eq. 12 may result in large error, and we may need another way to approximate the integral in Eq. 8.

When the nonlinear ICA algorithm has run for a certain number of epochs, $\mathbf{u}$, the output of the system in Figure 2, has approximately independent components and is approximately uniformly distributed in $[0,1]^n$. This means that $p(\mathbf{u})$ is approximately 1. As $p(\mathbf{x}) = p(\mathbf{u}) \cdot |\det \mathbf{J}|$, one can see that $p(\mathbf{x})$ is approximately equal to $|\det \mathbf{J}|$. Consequently Eq. 13 becomes $R_{local}(\theta) \approx E\Big\{\frac{1}{|\det \mathbf{J}|}\sum_{i=1}^{n}\sum_{j=1}^{n} P_{ij}\Big\}$. The gradient of $R_{local}(\theta)$ is

$$\frac{\partial R_{local}(\theta)}{\partial \theta} \approx E\Big\{\frac{1}{|\det \mathbf{J}|}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial P_{ij}}{\partial \theta}\Big\}. \tag{14}$$

As $\mathbf{J} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ can be easily calculated according to the network structure in Figure 2, Eq. 14 is also easy to evaluate, using Eq. 20 of Appendix B.

3.2 MND for Nonlinear ICA Based on Kernels

Nonlinear ICA based on kernels (Zhang and Chan, 2007a) exploits kernel methods to construct the separation system $\mathcal{G}$, and unknown parameters are adjusted by minimizing the mutual information between outputs y_i.[5] We have applied the MND principle and the smoothness regularizer to nonlinear ICA based on kernels; for details, see Zhang and Chan (2007a). Note that unlike the mapping provided by a MLP, which is comparatively smooth, the mapping constructed by kernel methods may not be smooth. So it is quite necessary to explicitly enforce the smoothness constraint for nonlinear ICA based on kernels.

4. Investigation of the Effect of MND

In this section we intend to explain why the MND principle, including the smoothness regularization, helps to alleviate the ill-posedness of nonlinear ICA from a mathematical viewpoint. There are two types of indeterminacies in solutions to nonlinear ICA, namely trivial indeterminacies and non-trivial indeterminacies. Trivial indeterminacies mean that the estimate of s_j produced by nonlinear ICA may be any nonlinear function of s_j; non-trivial indeterminacies mean that the outputs of nonlinear ICA, although mutually independent, are still a mixing of the original sources. Let us begin with the effect of MND on trivial indeterminacies.

4.1 For Trivial Indeterminacies

Let us assume in this section that, in the solutions of nonlinear ICA, each component depends only on one of the sources. Before presenting the main result, let us first give the following lemma.

Lemma 1 *Suppose that we are given the random vector* $\mathbf{d} = (d_1, d_2, \cdots, d_n)^T$. *Let* R_y *be the mean square error of reconstructing* $\mathbf{d}$ *from the variable* y *with the best-fitting linear transformation, that is,* $R_y = \min_{\mathbf{a}} E\{||\mathbf{d} - \mathbf{a} \cdot y||^2\}$, *where* $\mathbf{a} = (a_1, a_2, \cdots, a_n)^T$. *The variable* y *which gives the minimum* R_y *is the first non-centered principal component of* $\mathbf{d}$ *multiplied by a constant, and if* y *is constrained to be zero-mean, it is the first principal component of* $\mathbf{d}$ *multiplied by a constant.*

See Appendix C for a proof. Now let us consider a particular kind of nonlinear mixtures, in which each observed nonlinear mixture x_i is assumed to be generated by

$$x_i = f_{i1}(s_1) + f_{i2}(s_2) + \cdots + f_{in}(s_n), \tag{15}$$

where f_{ij} are invertible functions. We call such nonlinear mixtures distorted source (DS) mixtures, since each observation is a linear mixture of nonlinearly distorted sources. For this nonlinear mixing model, we have the following theorem on the effect of MND on trivial indeterminacies in its nonlinear ICA solutions. Here the following assumptions are made:

A1: In the output of nonlinear ICA, each component depends only on one of the sources and is zero-mean.

A2: The nonlinear ICA system has enough flexibility to reach the minimum of the MND regularization term R_{MSE} defined by Eq. 2.

5. The difference between nonlinear ICA based on kernels discussed here and the kernel-based nonlinear BSS method by Harmeling et al. (2003) should be made clear. Both of them use kernels. However, the former produces statistically independent outputs, while the latter exploits the temporal structure of the sources for separation.

Theorem 1 *Suppose that each observed nonlinear mixture x_i is generated according to Eq. 15. Under assumptions A1 & A2, the estimate of s_j produced by nonlinear ICA with MND is the first principal component of $\mathbf{f}_{*j}(s_j) = [f_{1j}(s_j), \cdots, f_{nj}(s_j)]^T$, multiplied by a constant.*

See Appendix D for a proof. The DS mixing model Eq. 15 may be restrictive. Now let us consider the case where nonlinearity in $\mathcal{F}$ is mild such that $\mathcal{F}$ can be well approximated by its Maclaurin expansion of degree 3. Let

$$\nabla_{i,j} = \frac{\partial x_i}{\partial s_j}\Big|_{s_j=0}, \nabla_{i,jk} = \frac{\partial^2 x_i}{\partial s_j \partial s_k}\Big|_{s_j,s_k=0}, \text{ and } \nabla_{i,jkl} = \frac{\partial^3 x_i}{\partial s_j \partial s_k \partial s_l}\Big|_{s_j,s_k,s_l=0}.$$

The following theorem discusses the effect of MND on trivial indeterminacies of nonlinear ICA solutions in this case. In particular, it states that by incorporating MND into the nonlinear ICA system, trivial indeterminacies in nonlinear ICA solutions are overcome; it shows how the outputs of the nonlinear ICA system, as the estimate of the sources, are related to the original sources s_i and the mixing system $\mathcal{F}$.

Theorem 2 *Suppose that each component of the mixing mapping $\mathcal{F} = (f_1, \cdots, f_n)^T$ in Eq. 1 is generated by the following Maclaurin series of degree 3:*

$$x_i = f_i(\mathbf{s}) = f_i(\mathbf{0}) + \sum_j \nabla_{i,j} s_j + \frac{1}{2}\sum_{j,k} \nabla_{i,jk} \cdot s_j s_k + \frac{1}{6}\sum_{j,k,l} \nabla_{i,jkl} \cdot s_j s_k s_l,$$

where $E\{s_j\} = 0$ and $E\{s_j^2\} = 1$, for $j = 1, \cdots, n$. Let

$$D_{ij}(s_j) \triangleq \left(\nabla_{i,j} + \frac{1}{2}\sum_{k\neq j} \nabla_{i,jkk}\right) \cdot s_j + \frac{1}{2}\nabla_{i,jj} \cdot s_j^2 + \frac{1}{6}\nabla_{i,jjj} \cdot s_j^3.$$

And let $\tilde{D}_{ij}(s_j)$ be the centered version of $D_i(s_j)$, that is, $\tilde{D}_{ij}(s_j) = D_{ij}(s_j) - E_i\{D_{ij}(s_j)\}$. Under assumptions A1 & A2, the estimate of s_j produced by nonlinear ICA with MND is the first principal component of $\tilde{\mathbf{D}}_{*j}(s_j) = [\tilde{D}_{1j}(s_j), \cdots, \tilde{D}_{nj}(s_j),]^T$, multiplied by a constant.

See Appendix E for a proof. Under the condition that nonlinear distortion in the mixing mapping $\mathcal{F}$ is not strong, $\tilde{D}_{ij}(s_j)$ would not be far from linear. Moreover, if the nonlinear part of $\tilde{D}_{ij}(s_j)$ varies for different i, the estimate of s_j is expected to be closer to linear than $\tilde{D}_{ij}(s_j)$, because it is the first principal component (PC) of $\tilde{\mathbf{D}}_{*j}(s_j)$. To summarize, Theorems 1 and 2 show that trivial indeterminacies in nonlinear ICA solutions can be overcome by the MND principle; and when the mixing mapping is not strong, the nonlinear distortion in the nonlinear ICA outputs w.r.t. the original sources is weak.

4.1.1 REMARK

In the proof of Theorems 1 and 2, we have made use of the fact that mutual information is invariant to any component-wise strictly monotonic nonlinear transformation of the variables. Consequently, trivial transformations do not affect the first term in Eq. 3, and they can be determined by minimizing R_{MSE} only, as claimed in the theorems. However, in practical implementations of nonlinear ICA algorithms, one needs to estimate the densities of y_i or their variations. Due to estimation error, the

gradient of the mutual information $I(y_1,\cdots,y_n)$ may be sensitive to the distribution of y_i, or it may be slightly affected by trivial transformations. This may cause the results of Theorems 1 and 2 to be violated slightly.

Fortunately, this phenomenon can be avoided easily. To model the trivial transformations, we apply a separate nonlinear function approximator (such as a MLP) to each output of nonlinear ICA to generate the final nonlinear ICA result. These nonlinear function approximators are then learned by minimizing R_{MSE} (Eq. 6). This provides a way to tackle the trivial indeterminacies; after performing nonlinear ICA with any nonlinear ICA method, if we know that there only exist trivial indeterminacies, we can adopt the above technique to determine the trivial transformations.

4.2 For Non-Trivial Indeterminacies

Now let us investigate the effect of MND on non-trivial indeterminacies in nonlinear ICA solutions. Generally speaking, there exist an infinite number of ways in which non-trivial indeterminacies occur, and it is impossible to formulate all of them. Hyvärinen and Pajunen (1999) gave some families of non-trivial transformations preserving mutual independence.

4.2.1 A PARTICULAR CLASS OF NON-TRIVIAL INDETERMINACIES

For the convenience of analysis, here we consider the following manner to construct non-trivial transformations preserving mutual independence. First, using the Gaussianization technique (Chen and Gopinath, 2001), we transform each of the independent variables s_i to a standard Gaussian variable u_i with an strictly increasing function q_i, that is, $u_i = q_i(s_i)$. Clearly u_i are mutually independent. Second, we can apply an orthogonal transformation $\mathbf{U}$ to $\mathbf{u} = (u_1,\cdots,u_n)^T$. The components of $\mathbf{e} = \mathbf{U}\mathbf{u}$ are still jointly Gaussian and mutually independent.[6] Finally, let $\mathbf{y} = \mathbf{r}(\mathbf{e})$, where $\mathbf{r} = (r_1,\cdots,r_n)^T$ is a component-wise function with each r_i strictly increasing. Components of $\mathbf{y}$ are still mutually independent. That is, $\mathbf{y}$ *is always a solution to nonlinear ICA of the nonlinear mixture* $\mathbf{x} = \mathcal{F}(s)$. The procedure transforming $\mathbf{s}$ to $\mathbf{y}$ can be described as $\mathbf{r}\circ\mathbf{U}\circ\mathbf{q}$, as shown in Figure 3. When $\mathbf{U}$ is a permutation matrix, this transformation is trivial; otherwise it is not.

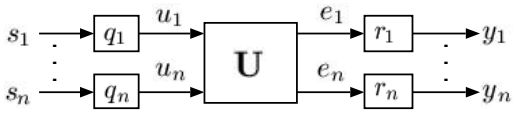


Figure 3: A non-trivial transformation from $\mathbf{s}$ to $\mathbf{y}$ preserving independence, that is, $\mathbf{r}\circ\mathbf{U}\circ\mathbf{q}$.

4.2.2 EFFECT OF MND

To see the effect of MND on $\mathbf{y}$ in Figure 3 (recall that $\mathbf{y}$ is a solution to nonlinear ICA of $\mathbf{x} = \mathcal{F}(\mathbf{s})$), we need to find how MND affects $\mathbf{U}$, as well as r_i. First, let us consider the case where the outputs y_i are Gaussian, meaning that each component of $\tilde{\mathbf{r}}$ is a linear mapping. Without loss of generality, we further assume that y_i are zero-mean and of unit variance, that is, $E\{\mathbf{y}\mathbf{y}^T\} = \mathbf{I}$. Consequently, r_i are identity mappings and $\mathbf{y} = \mathbf{e} = \mathbf{U}\mathbf{u}$. Assuming x_i are zero-mean, according to Eq. 5, We have $R_{MSE} = -\mathrm{Tr}\big(E\{\mathbf{x}\mathbf{y}^T\}E\{\mathbf{y}\mathbf{x}^T\}\big) + \mathrm{const} = -\mathrm{Tr}\big(E\{\mathbf{x}\mathbf{u}^T\}\mathbf{U}^T\mathbf{U}E\{\mathbf{u}\mathbf{x}^T\}\big) + \mathrm{const} = -\mathrm{Tr}\big(E\{\mathbf{x}\mathbf{u}^T\}E\{\mathbf{u}\mathbf{x}^T\}\big) + \mathrm{const}$. In this case R_{MSE} is not affected by $\mathbf{U}$, and the MND principle

6. $\mathbf{U}$ may depend on $||\mathbf{u}||$. In other words, $\mathbf{U}$ may be different for $\mathbf{u}$ of different norms.

could not help to avoid such non-trivial indeterminacies. We have empirically found that in general, when y_i are close to Gaussian, the separation performance tends to be bad. To make sure that the separation result is reliable, one should check the non-Gaussianity of y_i after the algorithm converges.

Next, suppose that that both s_l and y_j are non-Gaussian. r_i are then nonlinear. Consider the extreme case that the mixing mapping $\mathcal{F}$ is linear; in order to minimize R_{MSE} (Eq. 2), $\mathbf{U}$ in Figure 3 must be a permutation matrix. One can then image that if nonlinearity in $\mathcal{F}$ is weak enough, $\mathbf{U}$ in Figure 3 should be approximately a permutation matrix, meaning that the original sources $\mathbf{s}$ could be recovered.

However, if nonlinearity in $\mathcal{F}$ is strong, $\mathbf{U}$ may not be a permutation matrix, and non-trivial transformations from $\mathbf{s}$ to $\mathbf{y}$ may occur. This is actually quite natural. Consider the mixing mapping $\mathbf{x} = \mathcal{F}(\mathbf{s})$ which can be decomposed as a non-trivial transformation of $\mathbf{s}$ shown in Figure 3 (denote by $\mathbf{z}$ its output), followed by a nonlinear transformation $\mathbf{x} = \mathcal{F}_L(\mathbf{z})$ which is close enough to linear. In this situation, the output of nonlinear ICA with MND would be an estimate of $\mathbf{z}$, and if no additional knowledge of the mixing mapping is given, it is impossible to recover the original sources s_i.

Below we give an two-channel example to illustrate the relationship between R_{MSE} and the orthogonal matrix $\mathbf{U}$ when nonlinearity in $\mathbf{F}$ is strong. The two independent sources are a uniformly distributed signal and a super-Gaussian signal, and their scatter plot is given in Figure 9(a). The observations x_i, whose scatter plot is shown in Figure 4(a), are generated by applying a 2-3-2 MLP to the source signals. From this figure we can see that nonlinearity in the mixing procedure is comparatively strong. The orthogonal matrix $\mathbf{U}$ in Figure 3 is parameterized as $\mathbf{U} = [\cos(\alpha), -\sin(\alpha); \sin(\alpha), \cos(\alpha)]$. From Eq. 6 and Figure 3, one can see that R_{MSE} depends on α and r_i. For each value of α, r_i ($i = 1, 2$) are modelled by a 1-6-1 MLP and they are learned by minimizing R_{MSE}. Finally, $\min_{r_i} R_{MSE}$ is a function of α, with a period of 90 degrees, as plotted in Figure 4(b). In this example, α determined by the MND principle is about 11 degrees. It is not that close to zero, but it is still comparatively small and consequently the sources s_i are recovered approximately.

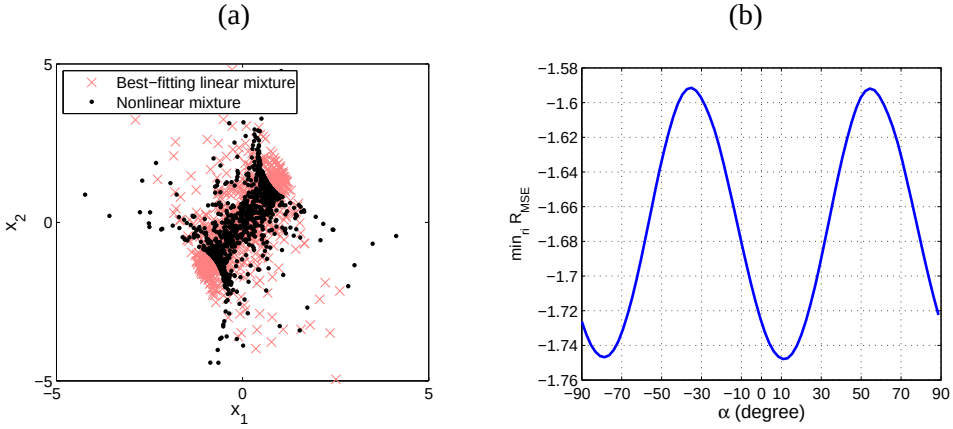


Figure 4: (a) Nonlinear mixtures of a sinusoid source signal and a super-Gaussian source signal (whose scatter plot is given in Figure 9.a) generated by a 2-3-2 MLP. x-mark points show linear mixtures of the sources which fit the nonlinear mixtures best. (b) $\min_{r_i} R_{MSE}$ as a function of α, whose minimum is achieved at $\alpha \approx 11$ degrees.

5. Simulations

In this section we investigate the performance of the proposed principle for solving nonlinear ICA using synthetic data. The experiments in Zhang and Chan (2007a) have empirically shown that both MND and the smoothness constraint are useful to ensure nonlinear ICA based on kernels to result in nonlinear BSS, when nonlinear distortion in the mixing procedure is not very strong. As its performance depends somewhat crucially on the choice of the kernel function, nonlinear ICA based on kernels is not used for comparison here. The following six methods (schemes) were used to separate various nonlinear mixtures:

1. MISEP: The MISEP method (Almeida, 2003) with parameters θ randomly initialized.[7] Note that in this method, the smoothness constraint has been implicitly incorporated to some extent, due to the property of the adopted MLP.

2. Linear init.: The MISEP method with $\mathcal{G}$ initialized as a linear mapping. This was achieved by adopting the regularization term Eq. 2 with $\lambda = 5$ (which is very large) in the first 50 epochs.

3. MND: The MISEP method incorporating MND, with R_{MSE}, the mean square error of the best linear reconstruction, as the regularization term (Section 2.2). The regularization parameter λ decayed from $\lambda_0 = 5$ to $\lambda_c = 0.01$ in the first 350 epochs. After that λ was fixed as λ_c.

4. Smooth (I): The MISEP method with the smoothness regularizer (Section 2.4) explicitly incorporated. λ decayed from 1 to 0.004 in the first 350 epochs.

5. Smooth (II): Same as Smooth (I), but λ was fixed to 0.007.

6. VB-NICA: Bayesian variational nonlinear ICA (Lappalainen and Honkela, 2000; Valpola, 2000).[8] PCA was used for initialization. After obtaining nonlinear factor analysis solutions using the package, we applied linear ICA (FastICA by Hyvärinen 1999 was used) to achieve nonlinear BSS.

In addition, in order to show the necessity of nonlinear ICA methods for separating nonlinear mixtures, linear ICA (FastICA was adopted) was also used to separate the nonlinear mixtures.

It was addressed in Section 2.3 that the incorporation of direct connections between inputs and output units in the MLP representing $\mathcal{G}$ implicitly and roughly implements the MND principle. To check that, in our experiments, the MLP without direct connections and that with direct connections were both adopted to represent $\mathcal{G}$, for comparison reasons. Like in Almeida (2003), the MLP has 20 arctan hidden units, 10 of which are connected to each of the output units of $\mathcal{G}$.

We use the signal to noise ratio (SNR) of y_i relative to s_i, denoted by $\mathrm{SNR}(y_i)$, to assess the separation performance of s_i. Besides, we apply a flexible nonlinear transformation h to y_i to minimize the MSE between $h(y_i)$ and s_i, and use the SNR of $h(y_i)$ relative to s_i as another performance measure. In this way possible trivial transformations between s_i and y_i are eliminated. In our experiments h was implemented by a two-layer MLP with eight hidden units with the hyperbolic tangent activation function and a linear output unit. This MLP was trained using the MATLAB neural network toolbox.

7. Source code is available at http : //www.lx.it.pt/ $\sim$ lbalmeida/ica/mitoolbox.html.
8. Source code is available at http : //www.cis.hut.fi/projects/bayes/. The following MATLAT commands were used to produce the ouput **y**: [nlfa_sources, net, params, status, fs] = nlfa(x, 'searchsources', 2, 'hidneurons', 15, 'iters', 2000); y = fastica(nlfa_sources.e, 'approach', 'symm', 'g', 'tanh');

Three kinds of nonlinear mixtures were investigated. They are distorted source (DS) mixtures, post-nonlinear (PNL) mixtures, and generic nonlinear (GN) mixtures which are generated by a MLP. Both super-Gaussian and sub-Gaussian sources were used.

5.1 For Distorted Source Mixtures

We first considered the DS mixtures defined in Eq. 15. Specifically, in the experiments the two-channel mixtures x_i were generated according to $x_1 = a_{11}s_1 + f_{12}(s_2)$, $x_2 = f_{21}(s_1) + a_{22}s_2$, where $a_{11} = a_{22} = 1$, and $f_{12}(s_i) = f_{21}(s_i) = 3\tanh(s_i/4) + 0.1s_i$. We used two super-Gaussian source signals, which are generated by $s_i = \frac{3}{5}n_i + \frac{2}{5}n_i^3$, where n_i are independent Gaussian signals. Each signal has 1000 samples. Figure 5 shows the scatter plot of the sources s_i and that of the observations x_i. To see the level of nonlinear distortion in the mixing transformation, we also give the scatter plot of the affine transformation of s_i which fits x_i the best.

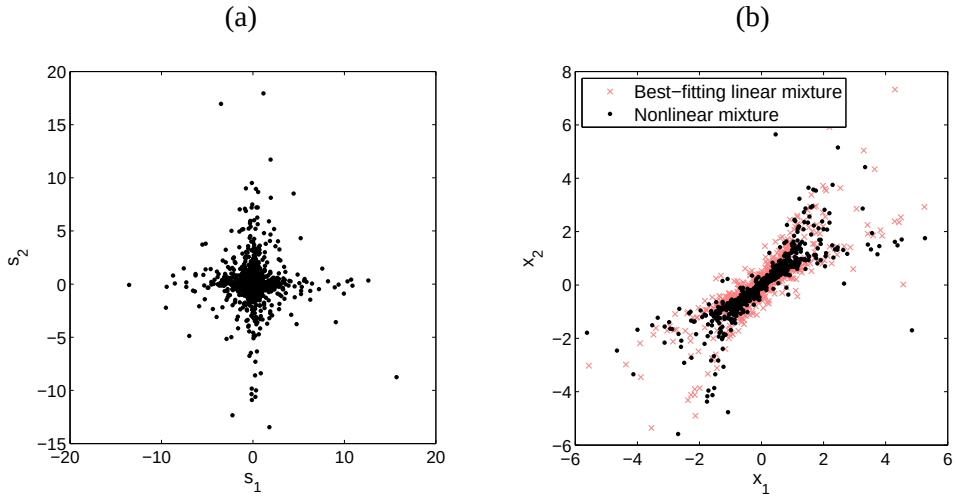


Figure 5: (a) Scatter plot of the sources s_i generating the DS mixtures. (b) Scatter plot of the DS mixtures x_i. x-mark points are linear mixtures of s_i which fit x_i best.

To reduce the random effect, all methods were repeated for 40 runs, and in each run the MLP was randomly initialized. We found that the separated results in the two channels have a similar SNR, so for saving space, here we just give the SNR in the first channel. Figure 6 compares the boxplot of SNR(y_1) and SNR$(h(y_1))$ for different methods. In Figure 6 (a, b), the MLP has no direct connections between inputs and output units, while in (c, d) the MLP has direct connections. We can see that in this case the methods MND, Smooth(I), and Smooth(II) give very high SNR, and at the same time, produce fewest unwanted results. Moreover, the MLP with direct connections behaves better than that without direct connections. The performance of VB-NICA is not very good. The reason may be that this method does not take into account the very useful information that nonlinearity in the mixing mapping is not very strong. It should be noted that VB-NICA may not exhibit its potential powerfulness in the experiments, since the source number is given and no noise is considered.

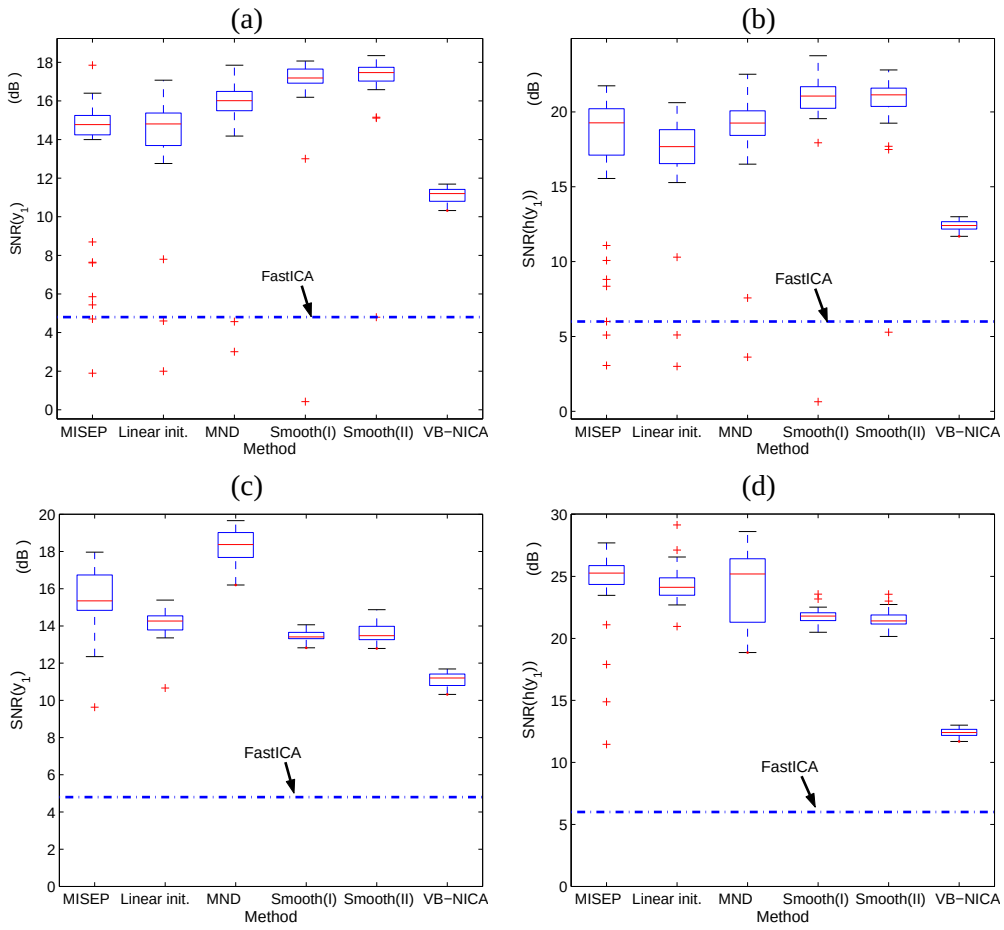


Figure 6: Boxplot of the SNR of separating the DS mixtures by the MLP without or with direct connections between inputs and output units. Top: Without direct connections. Bottom: With direct connections. (a, c) SNR(y_1). (b, d) SNR$(h(y_1))$.

5.2 For Post-Nonlinear Mixtures

The second experiment is to separate PNL mixtures. We used two sub-Gaussian source signals, which are a uniformly distributed white signal and a sinusoid waveform. The sources were first mixed with the mixing matrix $\mathbf{A} =$ [-0.2261, -0.1189; -0.1706, -0.2836], producing linear mixtures $\mathbf{z}$. The observations were then generated as $x_1 = z_1/2.5 + \tanh(3z_1)$ and $x_2 = z_2 + z_2^3/1.5$. Figure 7 shows the scatter plot of the sources and that of the PNL mixtures (after standardization). Figure 8 gives the separation performance of s_1 by various methods.[9] In this case, the proposed nonlinear ICA with MND (labelled by MND) also gives almost the best results; especially for the MLP without direct connections, the result of nonlinear ICA with MND is clearly the best. Again, the MLP with direct connections produces better results. Moreover, one can see that compared to the DS

9. If we use the PNL mixing model (Taleb and Jutten, 1999) to separate such mixtures, theoretically the sources could be well recovered. But in this paper we assume that the form of the mixing procedure is unknown, and treat it as a general nonlinear ICA problem.

mixtures in Section 5.1, the PNL mixtures considered here are comparatively hard to be separated by the MLP structure.

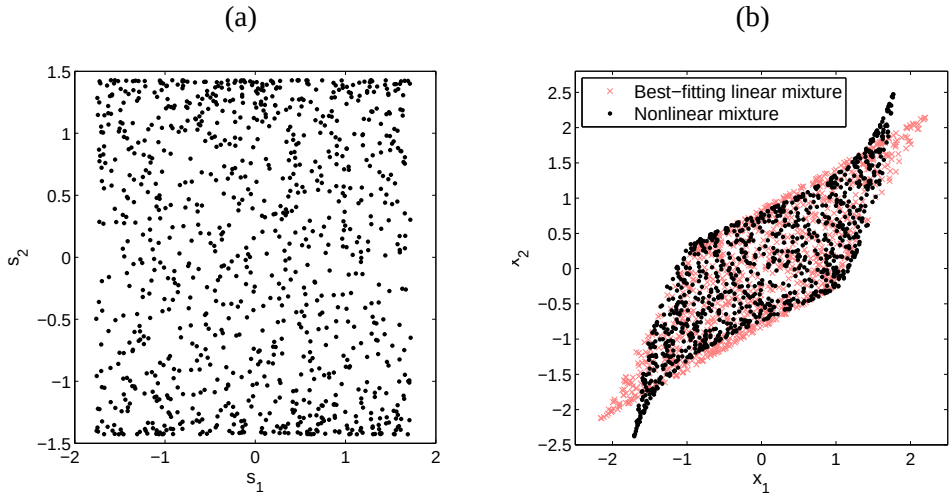


Figure 7: (a) Scatter plot of the sources s_i generating PNL mixtures. (b) Scatter plot of the PNL mixtures x_i.

5.3 For Generic Nonlinear Mixtures

We used a 2-2-2 MLP to generate nonlinear mixtures from sources. Hidden units have the arctan activation function. The weights between the input layer and the hidden layer are random numbers between -1 and 1. They are not large such that the mixing mapping is invertible and the nonlinear distortion produced by the MLP would not be very strong. The sources used here were the first source in Experiment 1 (super-Gaussian) and the second one in Experiment 2 (sub-Gaussian). Figure 9 shows the scatter plot of the sources and that of the GN mixtures. The performance of various methods for separating such mixtures is given in Figures 10. Apparently nonlinear ICA with MND gives the best separation results in this case.

Summed over all the three cases discussed above, we can see that MISEP with MND produces promising results for the general nonlinear ICA problem, provided that nonlinearity in the mixing mapping is not very strong. Specifically, it gives the fewest unwanted solutions, and its separation performance is very good. Moreover, the MLP with direct connections usually performs better than that without direct connections, but we also found that in some cases it got stuck into unwanted solutions more easily.

5.4 On Trivial Indeterminacies

In Section 4.1 we have discussed the effect of the MND principle on trivial indeterminacies of nonlinear ICA solutions. In particular, Theorem 1 states that for DS mixtures, if there are only trivial indeterminacies, each output of nonlinear ICA with MND is the PC of the contributions of the corresponding source to all mixtures. Now let us illustrate this with the help of the DS mixtures used in Section 5.1.

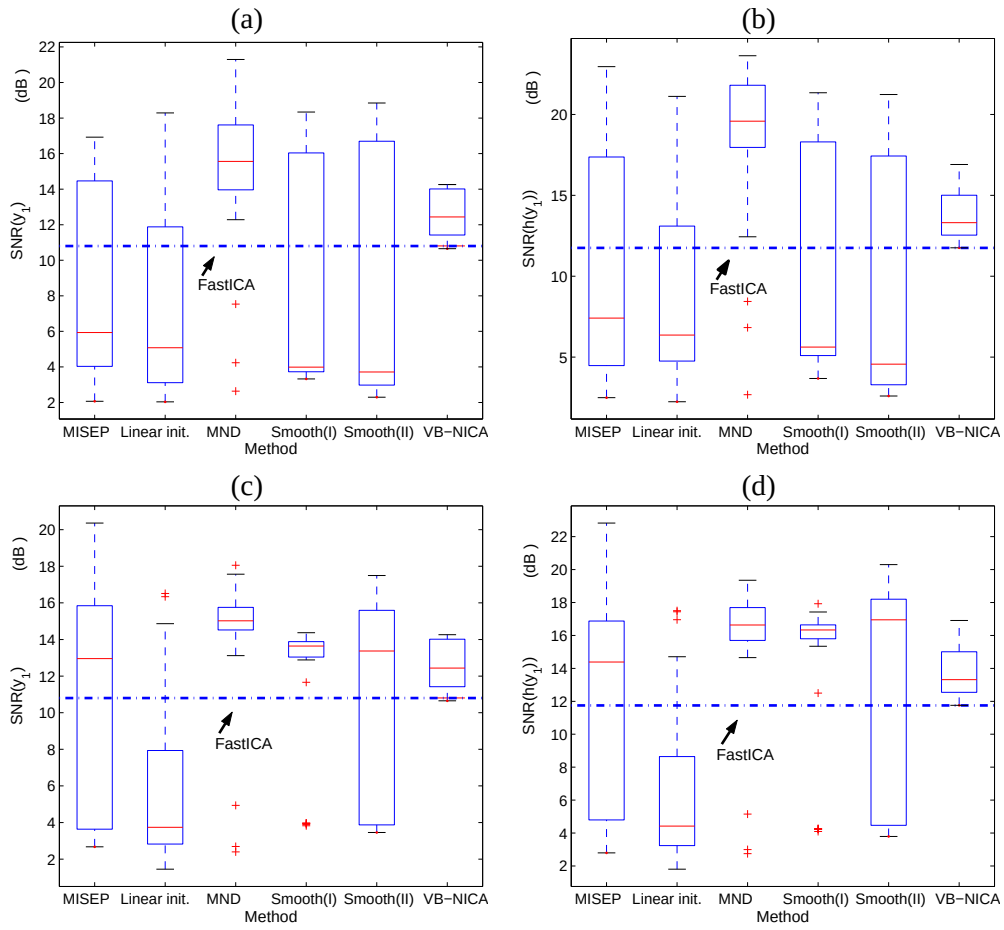


Figure 8: Boxplot of the SNR of separating the PNL mixtures by the MLP without or with direct connections between inputs and output units. Top: Without direct connections. Bottom: With direct connections. (a, c) SNR(y_1). (b, d) SNR$(h(y_1))$.

Figure 11 shows the relationship between y_i obtained by MISEP with MND in one run and the PC of $\mathbf{f}_{*i}(s_i) = [f_{1i}(s_i), f_{2i}(s_i)]^T$. We can see that each y_i is actually not very close to the corresponding PC, which may be caused by two reasons. First, there may exist some weak non-trivial transformation in the solution, as seen from the points close to the origin in Figure 11(b); y_2 is not solely dependent on s_2, but also slightly affected by s_1. The other reason is the error in estimating the density of y_i or its variation involved in the MISEP method, as explained in Section 4.1.1. We use the method proposed there to avoid the effect of the estimation error: a 1-8-1 MLP, denoted by τ_i, is applied to each y_i, and $\tau_i(y_i)$ is taken as the final nonlinear ICA output. Each τ_i is learned by minimizing R_{MSE} (Eq. 6). The resulting $\tau_i(y_i)$ is almost identical to the corresponding PC of $\mathbf{f}_{*i}(s_i)$, as seen from Figure 12. This has confirmed Theorem 1 and the validity of the method for tackling trivial indeterminacies proposed in Section 4.1.1.

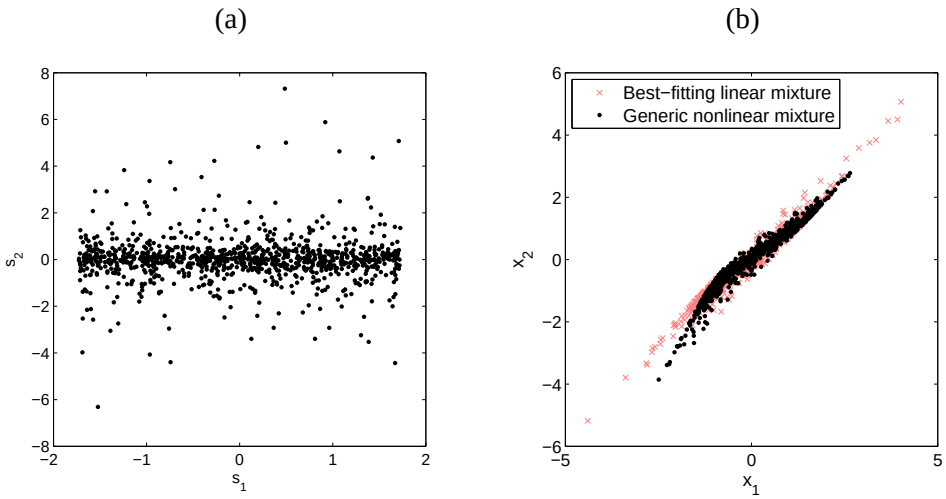


Figure 9: (a) Scatter plot of the sources s_i. (b) Scatter plot of the GN mixtures x_i.

6. Application to Causality Discovery in the Hong Kong Stock Market

In this section we give a real-life application of nonlinear ICA with MND. Specifically, we use this method to discover linear causal relations among the daily returns of a set of stocks. The empirical results were ever reported in Zhang and Chan (2006), without much detail of the method.

6.1 Introduction

It is well known that financial assets are not independent of each other, and that there may be some relations among them. Such relations can be described in different ways. In risk management, correlations are used to describe them and help to construct portfolios. The business group, which is a collection of firms bound together in some formal and/or informal ways, focuses on ties between financial assets and has attracted a lot of interest (Khanna and Rivkin, 2006). But these descriptions cannot tell us the causal relations among the financial assets.

The return of a particular stock may be influenced by those of other stocks, for many reasons, such as the ownership relations and financial interlinkages (Khanna and Rivkin, 2006). According to the efficient market hypothesis, such influence should be reflected in the stock returns immediately. In this part we aim to discover the causal relations among selected stocks by analyzing their daily returns.[10]

Traditionally, causality discovery algorithms for continuous variables usually assume that the dependencies are of a linear form and that the variables are Gaussian distributed (Pearl, 2000). Under the Gaussianity assumption, only the correlation structure of variables is considered and all higher-order information is neglected. As a consequence, one obtains some possible causal dia-

10. In other words, here we aim to find the "instantaneous" causality in the stock market. In contrast, Granger causality (Granger, 1980) analysis has become an important tool to find the "lagged" causality between time series. A time series x_1 "Granger causes" another series x_2 if by incorporating the past history of x_1 can improve a prediction of x_2 over a prediction based only on the history of x_2 alone. The efficient market hypothesis implies no significant Granger causality between stock returns. In fact, we have applied the approach by Reale and Tunnicliffe Wilson (2001) and partial directed coherence (Baccala and Sameshima, 2001) to find the Granger causality among the selected stocks, and very few Granger causal relations were found.

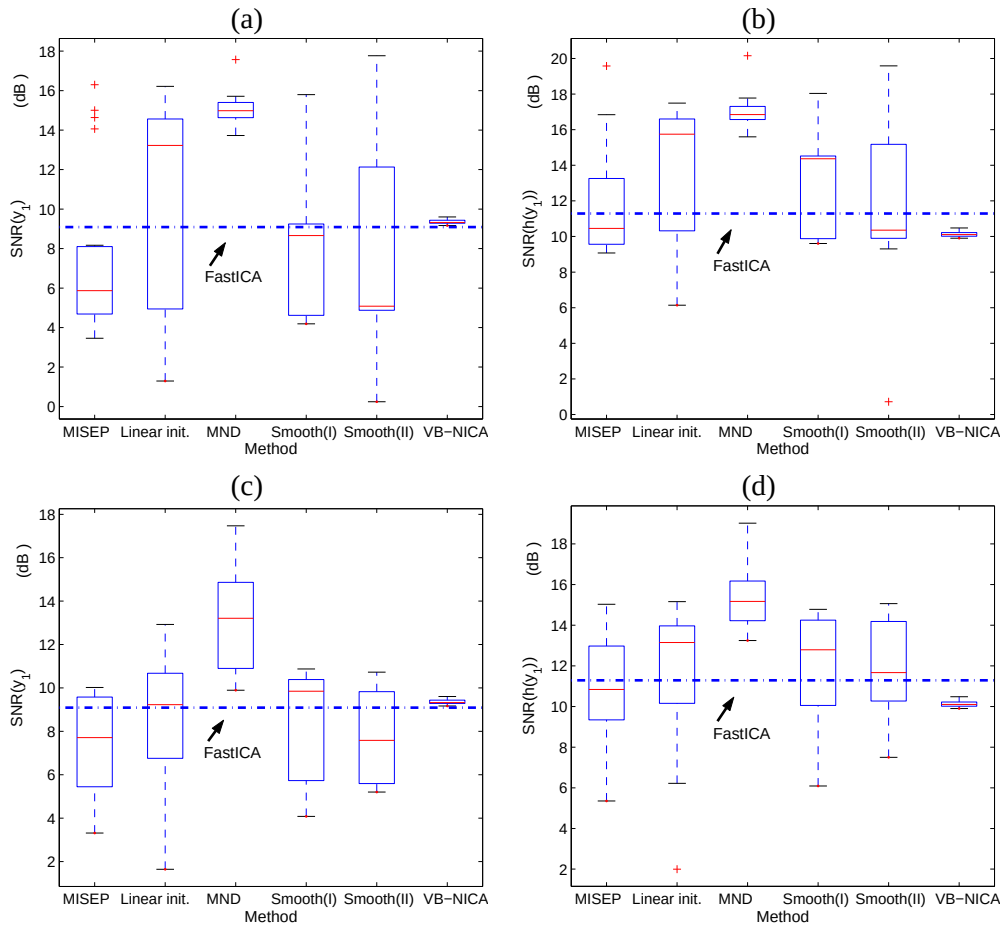


Figure 10: Boxplot of the SNR of separating the GN mixtures by the MLP without or with direct connections between inputs and output units. Top: Without direct connections. Bottom: With direct connections. (a, c) SNR(y_1). (b, d) SNR$(h(y_1))$.

grams which are equivalent in their correlation structure, and cannot find the true causal directions. Recently, it has been shown that the non-Gaussianity distribution of the variables allows us to distinguish the explanatory variable from the response variable, and consequently, to identify the full causal model (Dodge and Rousson, 2001; Shimizu et al., 2006).

In particular, in Shimizu et al. (2006) an elegant and efficient method was proposed for identifying the *linear, non-Gaussian, acyclic causal model* (abbreviated LiNGAM) by exploiting ICA. If the data are generated according to the LiNGAM model, theoretically, the ICA de-mixing matrix $\mathbf{W}$ can be permuted to lower triangularity. However, in practice, this may not hold, due to the finite sample effect, the existence of unobserved confounder variables (Pearl, 2000), or mild nonlinearity and noise that are often encountered in the data generation procedure. To tackle possible mild nonlinearity in the data generation procedure, we use nonlinear ICA with MND, instead of linear ICA, to separate the observed data. As the nonlinear distortion is mild, it can be neglected and consequently, linear causal relations among the observed data can be discovered.

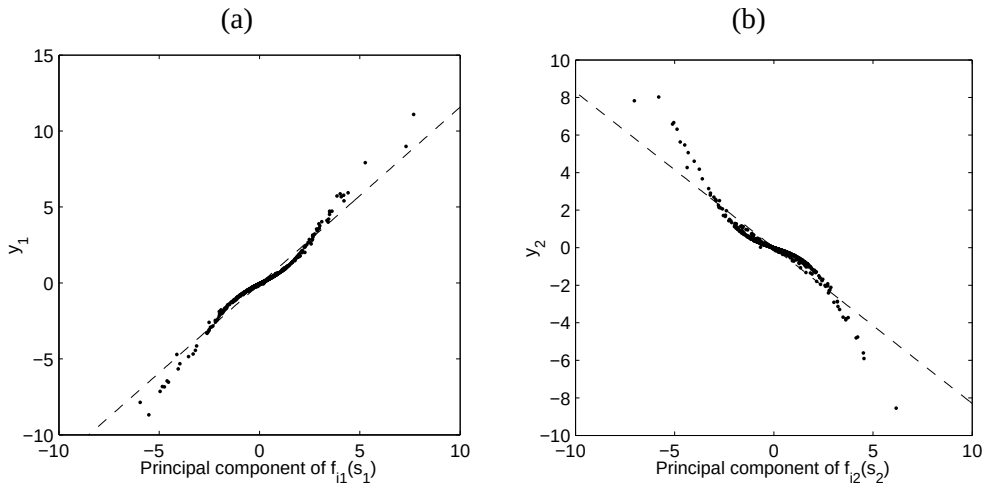


Figure 11: (a) y_1 recovered by MISEP with MND versus the PC of the contributions of s_1 to the DS mixtures used in Section 5.1. The SNR of y_1 w.r.t. the PC of the contributions of s_1 is 13.48dB. The dashed line is the linear function fitting the points best. (b) y_2 versus the PC of the contributions of s_2 to the DS mixtures. The SNR is 9.12dB.

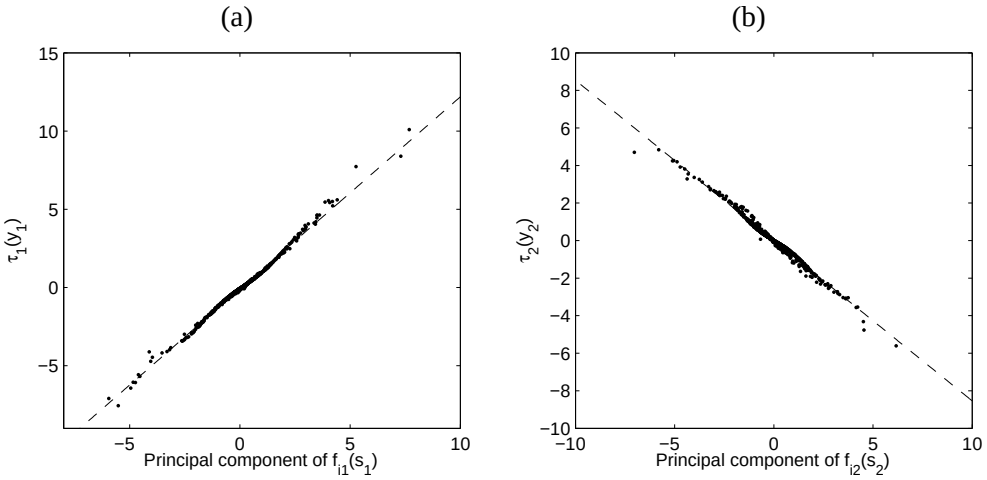


Figure 12: (a) $\tau_1(y_1)$ versus the PC of the contributions of s_1 to x_i. τ_1 is modelled by a 1-8-1 MLP and is learned by minimizing R_{MSE} (Eq. 6). The SNR is 20.99dB. (b) $\tau_2(y_2)$ versus the PC of the contributions of s_2 to x_i. The SNR is 18.64dB.

6.2 Causality Discovery by ICA: Basic Idea

The LiNGAM model assumes that the generation procedure of the observed data follows the following properties (Shimizu et al., 2006). *1.* It is recursive. This is, the observed variables x_i, $i = 1,...,n$, can be arranged in a causal order, such that no later variable causes any earlier variable. This causal order is denoted by $k(i)$. *2.* The value of x_i is a linear function of the values assigned to the earlier variables, plus a disturbance term e_i and an optional constant c_i: $x_i = \sum_{k(j)<k(i)} b_{ij} x_j + e_i + c_i$. *3.*

e_i are independent continuous-valued variables with non-Gaussian distributions (or at most one is Gaussian).

After centering of the variables, the causal relations among x_i can be written in the matrix form: $\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{e}$, where $\mathbf{x} = (x_1, ..., x_n)^T$, $\mathbf{e} = (e_1, ..., e_n)^T$, and the matrix $\mathbf{B}$ can be permuted (by simultaneous equal row and column permutations) to strict lower triangularity if one knows the causal order $k(i)$ of x_i. We then have $\mathbf{e} = \mathbf{W}\mathbf{x}$, where $\mathbf{W} = \mathbf{I} - \mathbf{B}$. This is exactly the ICA separation procedure (Hyvärinen et al., 2001). Therefore, the LiNGAM model can be estimated by ICA. We can permute the rows of the ICA de-mixing matrix $\mathbf{W}$ such that it produces a matrix $\widetilde{\mathbf{W}}$ without any zero on its diagonal (or in practice, $\sum_i |\widetilde{\mathbf{W}}_{ii}|$ is maximized). Dividing each row of $\widetilde{\mathbf{W}}$ by the corresponding diagonal entry gives a new matrix $\widetilde{\mathbf{W}}'$ with all entries on its diagonal equal to 1. Finally, by applying equal row and column permutations on $\mathbf{B} = \mathbf{I} - \widetilde{\mathbf{W}}'$, we can find the matrix $\widetilde{\mathbf{B}}$ which is as close as possible to strictly lower triangularity. $\widetilde{\mathbf{B}}$ contains the causal relations of x_i. For details, see Shimizu et al. (2006).

6.3 With Nonlinear ICA with Minimal Nonlinear Distortion

We now consider a general case of the nonlinear distortion often encountered in the data generation procedure, provided that the nonlinear distortion is smooth and mild. We use the MLP structure described in Section 3.1.1, which is a linear transformation coupled with an ordinary MLP, as shown in Figure 13, to model the nonlinear transformation from the the observed variables x_i to the disturbance variables e_i.

According to Figure 13, we have $\mathbf{e} = \mathbf{W}\mathbf{x} + \mathbf{h}(\mathbf{x})$, and consequently $\mathbf{x} = (\mathbf{I} - \mathbf{W})\mathbf{x} - \mathbf{h}(\mathbf{x}) + \mathbf{e}$, where $\mathbf{h}(\mathbf{x})$ denotes the output of the MLP. As it is difficult to analyze the relations among x_i implied by the nonlinear transformation $\mathbf{h}(\mathbf{x})$, we expect that $\mathbf{h}(\mathbf{x})$ is weak such that its effect can be neglected. The *linear* causal relations among x_i can then be discovered by analyzing $\mathbf{W}$.

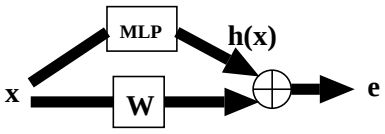


Figure 13: Structure used to model the transformation from the observed data x_i to independent disturbances e_i. $\mathbf{h}(\mathbf{x})$ accounts for nonlinear distortion if necessary.

In order to do causality discovery, the separation system in Figure 13 is expected to exhibit the following properties. *1*. The outputs e_i are mutually independent, since independence of e_i is a crucial assumption in LiNGAM. This can be achieved since nonlinear ICA always has solutions. *2*. The matrix $\mathbf{W}$ is sparse enough such that it can be permuted to lower triangularity. This can be enforced by incorporating the L_1 (Hyvärinen and Karthikesh, 2000) or smoothly clipped absolute deviation (SCAD) penalty (Fan and Li, 2001) on the entries of $\mathbf{W}$. *3*. The nonlinear mapping modeled by the MLP is weak enough such that we just care about the linear causal relations indicated by $\mathbf{W}$. To achieve that, we use MISEP with MND given in Section 3.1. In addition, we initialize the system with linear ICA results. That is, $\mathbf{W}$ is initialized by the linear ICA de-mixing matrix, and the initial values for weights in the MLP $\mathbf{h}(\mathbf{x})$ are very close to 0. The training process is terminated once the LiNGAM property holds for $\mathbf{W}$. After the algorithm terminates, $\frac{\text{var}(h_i(\mathbf{x}))}{\text{var}(e_i)}$ can be used to measure the level of nonlinear distortion in each channel, if needed.

6.4 Simulation Study

We examined the performance of the scheme discussed in Section 6.3 for identifying linear causal relations using simulated data. To make the nonlinear distortion in the data generation procedure weak, we used the structure in Figure 13 to generate the 8-dimensional observed data x_i from some independent and non-Gaussian variables e_i, that is, x_i are generated by a linear transformation coupled with a MLP.

The linear transformation in the data generation procedure was generated by $\mathbf{A} = (\mathbf{I} - \mathbf{B})^{-1}$. It satisfies the LiNGAM property since $\mathbf{B}$ was made strict lower triangular. The magnitude of non-zero entries of $\mathbf{B}$ is uniformly distributed between 0.05 and 0.5, and the sign is random. To examine if spurious causal relations would be caused, we also randomly selected 9 entries in the strict lower triangular part of $\mathbf{B}$ and set them to zero. The disturbance variables were obtained by passing independent Gaussian variables through power non-linearities with the exponent between 1.5 and 2. The variances of e_i were randomly chosen between 0.2 and 1. These settings are similar to those in the simulation studies by Shimizu et al. (2006). The sample size is 1000. The nonlinear part is a 8-10-8 MLP with the arctan activation function in the hidden layer. The weights from the inputs to the hidden layer are between -3 and 3, that is, they are comparatively large, while those from the hidden layer to the outputs are small, such that the nonlinear distortion is weak. The nonlinear distortion level in the generation procedure is measured by the ratio of the variance of the MLP output to that of the linear output. We considered two cases where the nonlinear distortion level is 0.01 and 0.03, respectively.

We used the scheme detailed in Section 6.3 to identify the linear causal relations among x_i. The SCAD penalty was used, and there are 10 arctan hidden units connected to each output of the MLP. We repeated the simulation for 100 trials. In each trial the maximum iteration number was set to 800. The results are given in Table 1 (numbers in parentheses are corresponding standard errors). The failure rate (the chance that LiNGAM does not hold for $\mathbf{W}$ within 800 iterations), the percentages of correctly identified non-zero edges, correctly identified large edges (with the magnitude larger than 0.2), and spurious edges in the successful cases, and the resulting nonlinear distortion level $\frac{\mathrm{var}(h_i(\mathbf{x}))}{\mathrm{var}(e_i)}$ in the separation system are reported. We can see that $\mathbf{W}$ almost always satisfies the LiNGAM property, and that most causal relations (especially large ones) are successfully identified. The setting $\lambda = 0.12$, meaning that MND is explicitly incorporated, gives better results than $\lambda = 0$ does, although the difference is not large. This is not surprising because even with $\lambda = 0$, nonlinear ICA with the separation structure of Figure 13 and with $\mathbf{W}$ initialized by linear ICA could achieve MND to some extent. However, when $\lambda = 0.12$, the nonlinear distortion in the separation system is much weaker, and we found that estimated values of the entries of $\mathbf{B}$ are closer to the true ones. The penalization parameter for SCAD, λ_{SCAD}, plays an important role. A larger λ_{SCAD} would make $\mathbf{W}$ satisfy the LiNGAM property more easily, but as a price, in the result more causal relations tend to disappear or be weaker.

For comparison, we also used linear ICA with the de-mixing matrix penalized by SCAD[11] for causality discovery. The result is reported in Table 2. Even when λ_{SCAD} is very large, which causes many causal relations to disappear, as seen from the table, there is still a high probability that the resulting de-mixing matrix fails to satisfy the LiNGAM property. These results show that for

11. The algorithm can be derived by maximizing the ICA likelihood penalized by the SCAD penalty on each entry of the de-mixing matrix. We used the natural gradient learning rule, with the score function adaptively estimated from the data.

Nonl. level in $\mathcal{F}$	Settings $(\lambda, \lambda_{SCAD})$	Fail. in 800 iter.	Edges identified	Large edges identified	Spurious edges	Nonl. level $\frac{\text{var}(h_i(\mathbf{x}))}{\text{var}(e_i)}$
0.01	$(0.12, 0.06)$	3%	88% (11%)	99% (3%)	7% (10%)	~ 0.03
	$(0, 0.06)$	3%	87% (12%)	97% (4%)	7% (9%)	~ 0.06
0.03	$(0.12, 0.10)$	1%	79% (14%)	92% (7%)	9% (11%)	~ 0.08
	$(0, 0.10)$	1%	76% (12%)	89% (8%)	8% (10%)	~ 0.13

Table 1: Simulation results of identifying linear causal relations among x_i with the nonlinear ICA structure Figure 13 and the SCAD penalty (100 trials). Numbers in parentheses are corresponding standard errors.

Nonl. level in $\mathcal{F}$	Settings	Fail. rate	Edges identified	Large edges identified	Spurious edges
0.01	$\lambda_{SCAD} = 0.2$	41%	67% (13%)	79% (15%)	4% (5%)
0.03	$\lambda_{SCAD} = 0.25$	54%	52% (12%)	58% (17%)	4% (7%)

Table 2: Simulation results of identifying linear causal relations among x_i by linear ICA with SCAD penalized de-mixing matrix (100 trials).

the data whose generation procedure has weak nonlinear distortion and approximately satisfies the LiNGAM property, nonlinear ICA with MND, together with the SCAD penalty, is useful to identify their linear causal relations.

6.5 Empirical Results

The Hong Kong stock market has some structural features different from the US and UK markets (Ho et al., 2004). One typical feature is the concentration of market activities and equity ownership in relatively small group of stocks, which probably makes causal relations in the Hong Kong stock market more obvious.

6.5.1 DATA

Here we aim at discovering the causality network among 14 stocks selected from the Hong Kong stock market.[12] The selected 14 stocks are constituents of Hang Seng Index (HSI).[13] They are almost the largest companies of the Hong Kong stock market. We used the daily dividend/split adjusted closing prices from Jan. 4, 2000 to Jun. 17, 2005, obtained from the Yahoo finance database. For the few days when the stock price is not available, we used simple linear interpolation to estimate the price. Denoting the closing price of the ith stock on day t by P_{it}, the corresponding return is calculated by $x_{it} = \frac{P_{it} - P_{i,t-1}}{P_{i,t-1}}$. The observed data are $\mathbf{x}_t = (x_{1t}, ..., x_{14,t})^T$. Each return series contains 1331 samples.

Recently ICA has been exploited as a possible way to explain the driving forces for financial returns (Back, 1997; Kiviluoto and Oja, 1998; Chan and Cha, 2001). We conjecture that nonlinear ICA would be more suitable than linear ICA to serve this task, since it seems reasonable that the

12. For saving space, they are not listed here; see the legend in Figure 15.

13. The only exception is Hang Lung Development Co. Ltd (0010.hk), which was removed from HSI on Dec. 2, 2002.

ICA mixing model varies slightly for returns at different levels. So we use nonlinear ICA with MND to analyze the stock returns and to do causality discovery. However, we should be aware that it is probably very hard to discover causal relations among the selected stocks, since the financial data are somewhat non-stationary, the data generation mechanism is not clear, and there may be many confounder variables.

6.5.2 RESULTS

We first applied a standard ICA algorithm to perform ICA on the data $\mathbf{x}_t$. The natural gradient algorithm (Amari et al., 1996) with the score function adaptively estimated from data was adopted. We used the LiNGAM software[14] to permute $\mathbf{W}$ and obtain the matrix $\mathbf{B} = \mathbf{I} - \widetilde{\mathbf{W}}'$. $\mathbf{B}$ seems unlikely to be lower-triangular; in fact, the ratio of the sum of squares of its upper-triangular entries to that of all entries is 0.24, which is very large. We also exploited linear ICA with the de-mixing matrix penalized by SCAD to do causality discovery. It was found that the learned de-mixing matrix $\mathbf{W}$ does not follow LiNGAM for $\lambda_{SCAD} \leq 0.25$. The value 0.25 for λ_{SCAD} is so large that statistical independence between outputs is affected. (In fact, most correlations between outputs have a magnitude larger than 0.1 when $\lambda_{SCAD} = 0.25$.) We may conclude that the data do not satisfy the LiNGAM model.

We then adopted the method proposed in Section 6.3. The SCAD penalty was applied to entries of $\mathbf{W}$ with $\lambda_{SCAD} = 0.04$. The regularization parameter for nonlinear ICA with MND (Eqs. 11 and 16–19) was $\lambda = 0.14$. After 195 epochs, $\mathbf{W}$ satisfies the LiNGAM assumption and the training process is terminated. Figure 14 shows the scatter plot of each output e_i and its linear part, from which we can see that the nonlinear distortion is weak. Based on the learned $\mathbf{W}$, we found the linear causal relations among these stocks, as shown in Figure 15. This figure was plotted using the LiNGAM software.

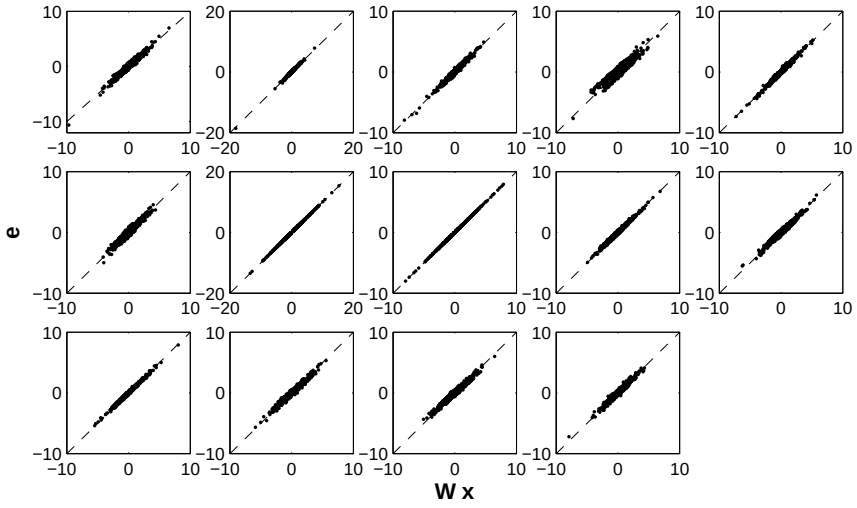


Figure 14: Scatter plot of each output of the system in Figure 13 and its linear part. The nonlinear distortion level $\frac{\mathrm{var}(h_i(\mathbf{x}))}{\mathrm{var}(e_i)}$ is 0.0485, 0.0145, 0.0287, 0.2075, 0.0180, 0.0753, 0, 0.0001, 0.0193, 0.0652, 0.0146, 0.0419, 0.0544, and 0.0492, respectively, for the 14 outputs e_i.

14. It is available at http : //www.cs.helsinki.fi/group/neuroinf/lingam/.

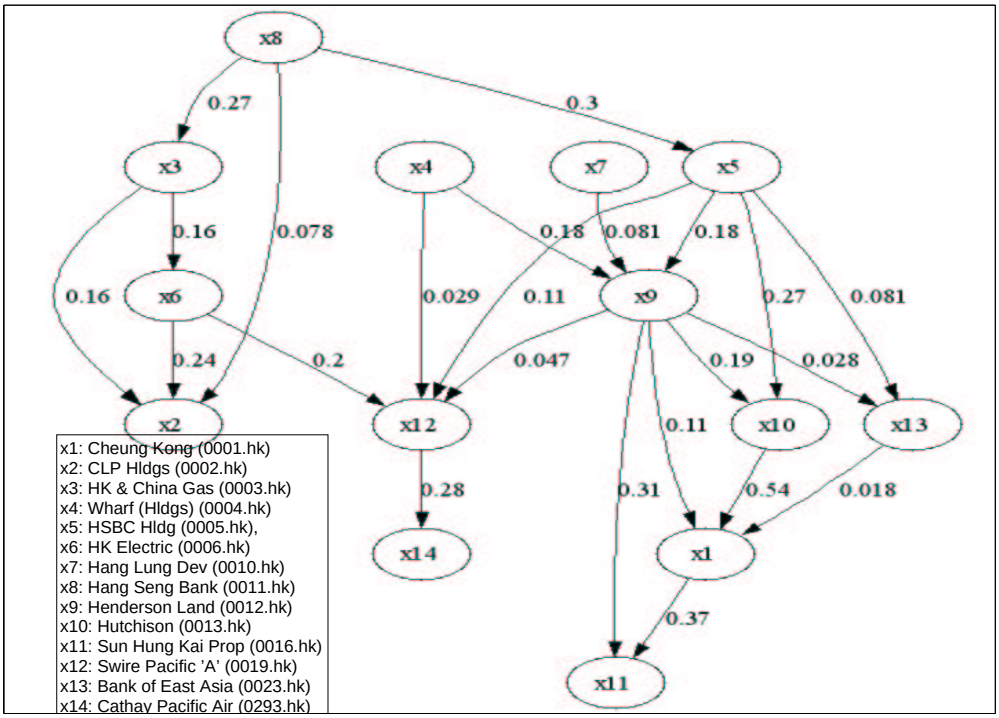


Figure 15: Causal diagram of the 14 stocks.

Figure 15 gives some interesting findings. *1*. Ownership relations tend to cause causal relations. If A is a holding company of B, there tends to be a causal relation from B to A. There are two significant relations $x_8 \to x_5$ and $x_{10} \to x_1$. In fact, x_5 owns some 60% of x_8, and x_1 holds about 50% of x_{10}. *2*. Stocks belonging to the same subindex tend to be connected together. For example, x_2, x_3, and x_6, which are linked together, are the only three constituents of Hang Seng Utilities Index. x_1, x_9, and x_{11} are constituents of Hang Seng Property Index. *3*. Large bank companies are the cause of many stocks. Here x_5 and x_8 are the two largest banks in Hong Kong. 4. Returns of stocks in Hang Seng Property Index tend to depend on many other stocks, while they hardly influence other stocks. Note that Here x_1, x_9, and x_{11} are in Hang Seng Property Index.

7. Conclusion

We have proposed the "minimal nonlinear distortion" principle to overcome the ill-posedness of the nonlinear ICA problem. With this principle, the nonlinear ICA solution whose estimated mixing system is close to linear would be preferred. This principle was implemented by a regularization technique that minimizes the mean square error of the best linear reconstruction of the observed mixtures. We explained how the proposed principle overcomes trivial and non-trivial indeterminacies in nonlinear ICA solutions. Experimental results on synthetic data in various situations showed that nonlinear ICA with minimal nonlinear distortion behaves very well and confirmed our theoretical claims. Since nonlinearity is usually encountered in practice and is not very strong in many cases, nonlinear ICA with minimal nonlinear distortion is expected to be capable of solving some real-life problems. Its successful application to causality discovery in the Hong Kong stock market illustrated the applicability of the method and the validity of the "minimal nonlinear distortion"

principle for some real problems. The result also supports the independent factor model in finance to some extent. Finally, it should be noted that solutions to nonlinear ICA or nonlinear BSS rely heavily on the prior information on the sources or the mixing mappings. "Minimal nonlinear distortion" is one type of such information for some problems. If more precise prior information, such as the form of the mixing mapping, the temporal structure of the sources, etc., is available, the separation result may be more meaningful.

Acknowledgments

This work was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administration Region, China. We are very grateful to the action editor and the anonymous referees for their valuable comments and suggestions. The first author would like to thank Haixuan Yang, Gang Li, and Wan Zhang for helpful discussions.

Appendix A. Gradient of R_{MSE}

Let $\mathbf{H} = diag\{h'(a_1), h'(a_2), ..., h'(a_M)\}$, and $\mathbf{W}_j^{(2)}$ denote the j-th column of $\mathbf{W}^{(2)}$. We have

$$\begin{aligned}
\frac{\partial R_{MSE}(\theta)}{\partial \mathbf{W}^{(1)}} &= E\Big\{\sum_{i=1}^{n} K_i \cdot \frac{\partial y_i}{\partial \mathbf{W}^{(1)}}\Big\} = E\Big\{\sum_{i=1}^{n} K_i \cdot \Big[\sum_{j=1}^{M} \frac{\partial y_i}{\partial a_j} \cdot \frac{\partial a_j}{\partial \mathbf{W}^{(1)}}\Big]\Big\} \\
&= E\Big\{\sum_{j=1}^{M} \Big[\Big(\frac{\partial \mathbf{y}}{\partial a_j}\Big)^T \mathbf{K}\Big] \cdot \frac{\partial a_j}{\partial \mathbf{W}^{(1)}}\Big\} = E\Big\{\sum_{j=1}^{M} \big[h'(a_j) \cdot \mathbf{W}_j^{(2)T} \cdot \mathbf{K}\big] \cdot \frac{\partial a_j}{\partial \mathbf{W}^{(1)}}\Big\} \\
&= E\{\mathbf{H} \cdot \mathbf{W}^{(2)T} \cdot \mathbf{K} \cdot \mathbf{x}^T\}, \qquad (16) \\
\frac{\partial R_{MSE}(\theta)}{\partial \mathbf{W}^{(d)}} &= E\Big\{\sum_{i=1}^{n} K_i \cdot \frac{\partial y_i}{\partial \mathbf{W}^{(d)}}\Big\} \\
&= E\{\mathbf{K}\mathbf{x}^T\}, \qquad (17) \\
\frac{\partial R_{MSE}(\theta)}{\partial \mathbf{b}^{(2)}} &= E\{\mathbf{K}\}, \qquad (18) \\
\frac{\partial R_{MSE}(\theta)}{\partial \mathbf{b}^{(1)}} &= E\{\mathbf{H} \cdot \mathbf{W}^{(2)T} \cdot \mathbf{K}\}. \qquad (19)
\end{aligned}$$

Appendix B. Gradient of P_{ij} in Eq. 8

Noting that $\frac{\partial}{\partial\theta}\left(\frac{\partial^2 y_l}{\partial x_i \partial x_j}\right) = \frac{\partial^2}{\partial x_i \partial x_j}\left(\frac{\partial y_l}{\partial \theta}\right)$ since θ is independent from x_i, we can obtain the following rule after tedious derivation:

$$\begin{aligned}
\frac{P_{ij}}{\partial w_{lm}^{(2)}} &= \frac{\partial^2 y_l}{\partial x_i \partial x_j} \cdot \frac{\partial^2 z_m}{\partial x_i \partial x_j}, \\
\frac{\partial P_{ij}}{\partial w_{mk}^{(1)}} &= \Delta_{ijm} \cdot \left\{ h''(a_m)[w_{mi}^{(1)} \cdot \delta_{kj} + w_{mj}^{(1)} \cdot \delta_{ik}] + h'''(a_m) \cdot w_{mj}^{(1)} \cdot w_{mi}^{(1)} \cdot x_k \right\}, \\
\frac{\partial P_{ij}}{\partial b_m^{(1)}} &= \Delta_{ijm} \cdot h'''(a_m) \cdot w_{mj}^{(1)} \cdot w_{mi}^{(1)}, \\
\frac{\partial P_{ij}}{\partial \mathbf{W}^{(d)}} &= \mathbf{0}, \\
\frac{\partial P_{ij}}{\mathbf{b}^{(2)}} &= \mathbf{0},
\end{aligned} \tag{20}$$

where $\Delta_{ijm} = \sum_{l=1}^{n} w_{lm}^{(2)} \cdot \frac{\partial^2 y_l}{\partial x_i \partial x_j}$, $\frac{\partial^2 y_l}{\partial x_i \partial x_j} = \sum_{m=1}^{M} w_{lm}^{(2)} \cdot \frac{\partial^2 z_m}{\partial x_i \partial x_j}$, $\frac{\partial^2 z_m}{\partial x_i \partial x_j} = h''(a_m) \cdot w_{mi}^{(1)} \cdot w_{mj}^{(1)}$, and δ_{ik} is the Kronecker delta function.

Appendix C. Proof of Lemma 1

Proof. The mean square error of reconstructing $\mathbf{d}$ from y with the linear transformation $\mathbf{a}$ is

$$\begin{aligned}
& E\{||\mathbf{d} - \mathbf{a} \cdot y||^2\} = E\{(\mathbf{d} - \mathbf{a} \cdot y)^T (\mathbf{d} - \mathbf{a} \cdot y)\} \\
= \; & E\{\mathbf{d}^T \cdot \mathbf{d} - 2\mathbf{a}^T \mathbf{d} \cdot y + \mathbf{a}^T \mathbf{a} \cdot y^2\} \\
= \; & E\left\{ \mathbf{a}^T \mathbf{a} \cdot \left(y - \frac{\mathbf{a}^T \mathbf{d}}{\mathbf{a}^T \mathbf{a}} \right)^2 - \frac{(\mathbf{a}^T \mathbf{d})^2}{\mathbf{a}^T \mathbf{a}} + \mathbf{d}^T \mathbf{d} \right\} \\
= \; & \mathbf{a}^T \mathbf{a} \cdot E\left\{ \left(y - \frac{\mathbf{a}^T \mathbf{d}}{\mathbf{a}^T \mathbf{a}} \right)^2 \right\} - E\left\{ \frac{(\mathbf{a}^T \mathbf{d})^2}{\mathbf{a}^T \mathbf{a}} \right\} + E\{\mathbf{d}^T \mathbf{d}\}.
\end{aligned} \tag{21}$$

The first term of Eq. 21 is always non-negative. No matter what value $\mathbf{a}$ takes, in order to minimize Eq. 21, we should choose

$$y = \mathbf{a}^T \mathbf{d} \cdot (\mathbf{a}^T \mathbf{a})^{-1} \tag{22}$$

to make this term vanish, meaning that y is the linear combination of d_i with the coefficients $\mathbf{a} \cdot (\mathbf{a}^T \mathbf{a})^{-1}$.

Next, when the first term of Eq. 21 vanishes, minimizing this function w.r.t. $\mathbf{a}$ is reduced to maximizing $E\{(\mathbf{a}^T \mathbf{d})^2 \cdot (\mathbf{a}^T \mathbf{a})^{-1}\} = E\{\mathbf{a}^T \mathbf{d}\mathbf{d}^T \mathbf{a} \cdot (\mathbf{a}^T \mathbf{a})^{-1}\}$. Letting $\mathbf{a}' = \mathbf{a}/\sqrt{\mathbf{a}^T \mathbf{a}}$, this is equivalent to the constrained optimization problem: $\max \mathbf{a}'^T \cdot E\{\mathbf{d}\mathbf{d}^T\} \cdot \mathbf{a}'$, s.t. $\mathbf{a}'^T \mathbf{a}' = 1$. Clearly this is the PCA problem. So $\mathbf{a}'$ is the eigenvector of $E\{\mathbf{d}\mathbf{d}^T\}$ associated with the largest eigenvalue, and according to Eq. 22, y is the principal component of $\mathbf{d}$ multiplied by a constant.

Now let us consider the case where y is constrained to be zero-mean. Let $\overline{\mathbf{d}} = E\{\mathbf{d}\}$, and $\tilde{\mathbf{d}} = \mathbf{d} - \overline{\mathbf{d}}$. We have $E\{||\mathbf{d} - \mathbf{a} \cdot y||^2\} = E\{(\tilde{\mathbf{d}} - \mathbf{a} \cdot y + \overline{\mathbf{d}})^T (\tilde{\mathbf{d}} - \mathbf{a} \cdot y + \overline{\mathbf{d}})\} = E\{(\tilde{\mathbf{d}} - \mathbf{a} \cdot y)^T (\tilde{\mathbf{d}} - \mathbf{a} \cdot y)\} + \overline{\mathbf{d}}^T \overline{\mathbf{d}}$. $\overline{\mathbf{d}}^T \overline{\mathbf{d}}$ can be considered as a constant. Using the result above, we can see that when R_y is minimized, y is the principal component of $\tilde{\mathbf{d}}$ multiplied by a constant. (Q.E.D)

Appendix D. Proof of Theorem 1

Proof: As it has been assumed here that each output of nonlinear ICA depends only on one of the sources, we can denote by $h_j(s_j)$ the estimate of s_j produced by nonlinear ICA. For the sake of simplicity, we make both x_i and $h_j(s_j)$ zero-mean, that is, $E\{x_i\} = E\{h_j(s_j)\} = 0$. So the matrix $\mathbf{A}^*$ in Eq. 2 is $n \times n$. Denote by a^*_{ij} the (i,j)th entry of $\mathbf{A}^*$. R_{MSE} defined by Eq. 2 is

$$\begin{aligned} R_{MSE} &= \sum_i E\Big\{x_i - \sum_j a^*_{ij} h_j(s_j)\Big\}^2 = \sum_i E\Big\{\sum_j \Big[f_{ij}(s_j) - a^*_{ij} h_j(s_j)\Big]\Big\}^2 \\ &= \sum_i \Big\{\sum_j E\big(f_{ij}(s_j) - a^*_{ij} h_j(s_j)\big)^2 \\ &\qquad + \sum_{k\neq l} E\big[\big(f_{ik}(s_k) - a^*_{ik} h_k(s_k)\big)\cdot\big(f_{il}(s_l) - a^*_{il} h_l(s_l)\big)\big]\Big\}. \end{aligned}$$

As $E\{h_k(s_k)h_l(s_l)\} = E\{h_k(s_k) f_{il}(s_l)\} = 0$ for $k \neq l$, the above equation becomes

$$\begin{aligned} R_{MSE} &= \sum_i \Big\{\sum_j E\big(f_{ij}(s_j) - a^*_{ij} h_j(s_j)\big)^2 + \sum_{k\neq l} E\big(f_{ik}(s_k) f_{il}(s_l)\big)\Big\} \\ &= \sum_j \Big\{\sum_i E\big(f_{ij}(s_j) - a^*_{ij} h_j(s_j)\big)^2\Big\} + \text{const.} \end{aligned}$$

One can see that minimization of the above function can be achieved by minimizing $\sum_i E\big(f_{ij}(s_j) - a^*_{ij} h_j(s_j)\big)^2$ independently for each j. That is, $h_j(s_j)$ and a^*_{ij} are adjusted to minimize $\sum_i E\big(f_{ij}(s_j) - a^*_{ij} h_j(s_j)\big)^2$. According to Lemma 1, $h_j(s_j)$ produced by nonlinear ICA with MND is the first principal component of $\mathbf{f}_{*j}(s_j) = [f_{1j}(s_j), \cdots, f_{nj}(s_j)]^T$, multiplied by a constant. (Q.E.D)

Appendix E. Proof of Theorem 2

Proof. Denote by $h_j(s_j)$ the estimate of s_j produced by nonlinear ICA, and assume that both x_i and $h_j(s_j)$ zero-mean. Denote by a^*_{ij} the (i,j)th entry of $\mathbf{A}^*$. Note that $\sum_{j,k,l} \nabla_{i,jkl} \cdot s_j s_k s_l = \sum_j \nabla_{i,jjj} \cdot s_j s_j s_j + 3 \cdot \sum_j \sum_{k\neq j} \nabla_{i,jkk} \cdot s_j s_k^2 + \sum_j \sum_{k\neq j} \sum_{\substack{l\neq j \\ l\neq k}} \nabla_{i,jkl} \cdot s_j s_k s_l$, and that $E\{s_j\} = 0$ and $E\{s_j^2\} = 1$. R_{MSE} defined by Eq. 2 becomes

$$\begin{aligned} & R_{MSE} = \sum_i E\Big\{x_i - \sum_j a^*_{ij} h_j(s_j)\Big\}^2 \\ = \; & \sum_i E\Big\{f_i(\mathbf{0}) + \sum_j \nabla_{i,j}\cdot s_j + \frac{1}{2}\sum_{j,k} \nabla_{i,jk}\cdot s_j s_k + \frac{1}{6}\sum_{j,k,l} \nabla_{i,jkl}\cdot s_j s_k s_l - \sum_j a^*_{ij} h_j(s_j)\Big\}^2 \\ = \; & \sum_{i=1}^n E\Big\{f_i(\mathbf{0}) + \sum_j \Big[\nabla_{i,j}\cdot s_j + \frac{1}{2}\nabla_{i,jj}\cdot s_j^2 + \frac{1}{6}\nabla_{i,jjj}\cdot s_j^3 + \frac{3}{6}\sum_{k\neq j} \nabla_{i,jkk}\cdot s_j s_k^2 \\ & - a^*_{ij} h_j(s_j)\Big] + \frac{1}{2}\sum_j \sum_{k\neq j} \nabla_{i,jk}\cdot s_j s_k + \frac{1}{6}\sum_j \sum_{k\neq j} \sum_{\substack{l\neq j \\ l\neq k}} \nabla_{i,jkl}\cdot s_j s_k s_l\Big\}^2. \end{aligned} \tag{23}$$

Bearing in mind that s_j are mutually independent, and also taking all the terms independent of $h_j(s_j)$ and a_{ij}^* as constants, we can re-write Eq. 23 as

$$\begin{aligned}
& R_{MSE} \\
= & \sum_i E\Big\{ \sum_j \Big[\nabla_{i,j} \cdot s_j + \frac{1}{2} \nabla_{i,jj} \cdot s_j^2 + \frac{1}{6} \nabla_{i,jjj} \cdot s_j^3 + \frac{3}{6} \sum_{k \neq j} \nabla_{i,jkk} \cdot s_j s_k^2 - a_{ij}^* h_j(s_j) \Big] \Big\}^2 + \text{const} \\
= & \sum_i E\Big\{ \sum_j \Big[\nabla_{i,j} \cdot s_j + \frac{1}{2} \nabla_{i,jj} \cdot s_j^2 + \frac{1}{6} \nabla_{i,jjj} \cdot s_j^3 - a_{ij}^* h_j(s_j) \Big] + \frac{1}{2} \sum_j \sum_{k \neq j} \nabla_{i,jkk} \cdot s_j s_k^2 \Big\}^2 + \text{const} \\
= & \sum_i E\Big\{ \Big[\sum_j \Big(\nabla_{i,j} \cdot s_j + \frac{1}{2} \nabla_{i,jj} \cdot s_j^2 + \frac{1}{6} \nabla_{i,jjj} \cdot s_j^3 - a_{ij}^* h_j(s_j) \Big) \Big]^2 \\
& - \sum_j \Big(a_{ij}^* h_j(s_j) \cdot \sum_{k \neq j} \nabla_{i,jkk} \cdot s_j s_k^2 \Big) \Big\} + \text{const} \\
= & \sum_i E\Big\{ \sum_j \Big(\nabla_{i,j} \cdot s_j + \frac{1}{2} \nabla_{i,jj} \cdot s_j^2 + \frac{1}{6} \nabla_{i,jjj} \cdot s_j^3 - a_{ij}^* h_j(s_j) \Big)^2 \\
& - \sum_j \Big(a_{ij}^* h_j(s_j) \cdot \sum_{k \neq j} \nabla_{i,jkk} \cdot s_j \Big\} + \text{const} \\
= & \sum_i \sum_j E\Big\{ \Big(\nabla_{i,j} + \frac{1}{2} \sum_{k \neq j} \nabla_{i,jkk} \Big) \cdot s_j + \frac{1}{2} \nabla_{i,jj} \cdot s_j^2 + \frac{1}{6} \nabla_{i,jjj} \cdot s_j^3 - a_{ij}^* h_j(s_j) \Big\}^2 + \text{const} \\
= & \sum_j \Big[\sum_i E\big(D_{ij}(s_j) - a_{ij}^* h_j(s_j) \big)^2 \Big] + \text{const.}
\end{aligned}$$

Note that there is no dependence relationship between $h_j(\cdot)$, as well as a_{ij}^*, with different j. To minimize the above function, we just need to adjust $h_j(s_j)$ and a_{ij}^* to minimize $\sum_i E\big(D_i(s_j) - a_{ij}^* h_j(s_j)\big)^2$, independently for each j. According to Lemma 1, $h_j(s_j)$ is the first principal component of $\tilde{\mathbf{D}}_{*j}(s_j) = [\tilde{D}_{1j}(s_j), \cdots, \tilde{D}_{nj}(s_j),]^T$, multiplied by a constant. (Q.E.D)

References

L.B. Almeida. MISEP - linear and nonlinear ICA based on mutual information. *Journal of Machine Learning Research*, 4:1297–1318, 2003.

L.B. Almeida. Separating a real-life nonlinear image mixture. *Journal of Machine Learning Research*, 6:1199–1229, 2005.

S. Amari, A. Cichocki, and H.H. Yang. A new learning algorithm for blind signal separation. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 757–763, Cambridge, MA, 1996. MIT Press.

L.A. Baccala and K. Sameshima. Partial directed coherence: a new concept in neural structure determination. *Biol. Cybern.*, 84:463–474, 2001.

A.D. Back. A first application of independent component analysis to extracting structure from stock returns. *International Journal of Neural Systems*, 8(4):473–484, August 1997.

A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

C.M. Bishop. Curvature-driven smoothing: a learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 4(5):882–884, 1993.

C.M. Bishop. Regularization and complexity control in feed-forward networks. In *Proc. International Conference on Artificial Neural Networks (ICANN'95)*, volume 1, pages 141–148, 1995.

G. Burel. Blind separation of sources: a nonlinear neural algorithm. *Neural Networks*, 5(6):937–947, 1992.

J.F. Cardoso. Blind signal separation: Statistical principles. *Proceeding Of The IEEE, special issue on blind identification and estimation*, 9(10):2009–2025, 1998.

L. Chan and S.M. Cha. Selection of independent factor model in finance. In *proceedings of 3rd International Conference on Independent Component Analysis and blind Signal Separation*, San Diego, California, USA, December 2001.

S.S. Chen and R.A. Gopinath. Gaussianization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 423–429. MIT Press, 2001.

A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. John Wiley & Sons, UK, corrected and revisited edition, 2003.

Y. Dodge and V. Rousson. On asymmetric properties of the correlation coefficient in the regression setting. *The American Statistician*, 55(1):51–54, 2001.

J. Eriksson and V. Koivunen. Blind identification of class of nonlinear instantaneous ICA models. In *Proc. of the XI European SIgnal Proc. Conf. (EUSIPCO 2002)*, volume 2, pages 7–10, Toulouse, France, Sept. 2002.

J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Amer. Statist. Assoc.*, 96:1348–1360, 2001.

C. Granger. Testing for causality: A personal viewpoint. *Journal of Economic Dynamics and Control*, 2:329-352, 1980.

W.E.L. Grimson. A computational theory of visual surface interpolation. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 298:395–427, 1982.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003.

R.Y. Ho, R. Strange, and J. Piesse. The structural and institutional features of the Hong Kong stock market: Implications for asset pricing. Research Paper 027, The Management Centre Research Papers, King's College London, 2004.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.

A. Hyvärinen and R. Karthikesh. Sparse priors on the mixing matrix in independent component analysis. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 477–452, Helsinki, Finland, 2000.

A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, Inc, 2001.

C. Jutten and A. Taleb. Source separation: From dusk till dawn. In *2nd International Workshop on Independent Component Analysis and Blind Signal Separation (ICA 2000)*, pages 15–26, Helsinki, Finland, 2000.

A.M. Kagan, Y.V. Linnik, and C.R. Rao. *Characterization Problems in Mathematical Statistics*. Wiley, New York, 1973.

T. Khanna and J.W. Rivkin. Interorganizational ties and business group boundaries: Evidence from an emerging economy. *Organization Science*, 17(3):333-352, 2006.

K. Kiviluoto and E. Oja. Independent component analysis for parallel financial time series. In *Proc. ICONIP'98*, volume 2, pages 895–898, Tokyo, Japan, 1998.

H. Lappalainen and A. Honkela. Bayesian nonlinear independent component analysis by multilayer perceptron. In M.Girolami, editor, *Advances in Independent Component Analysis*, pages 93–121. Spring-Verlag, 2000.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000.

T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317: 314–319, 1985.

M. Reale and G. Tunnicliffe Wilson. Identification of vector ar models with recursive structural errors using conditional independence graphs. *Statistical Methods and Applications*, 10(1-3): 49–65, 2001.

S. Shimizu, P.O. Hoyer, A. Hyvärinen, and A.J. Kerminen. A linear non-Gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7:2003–2030, 2006.

A. Taleb. A generic framework for blind source separation in structured nonlinear models. *IEEE Transactions on Signal Processing*, 50(8):1819–1830, 2002.

A. Taleb and C. Jutten. Source separation in post-nonlinear mixtures. *IEEE Trans. on Signal Processing*, 47(10):2807–2820, 1999.

Y. Tan, J. Wang, and J. M. Zurada. Nonlinear blind source separation using a radial basis function network. *IEEE Trans. on Neural Networks*, 12(1):124–134, 2001.

A.N. Tikhonov and V.A. Arsenin. *Solutions of Ill-posed Problems*. Winston & Sons, Washington, 1977.

H. Valpola. Nonlinear independent component analysis using ensemble learning: Theory. In *Proc. Int. Workshop on Independent Component Analysis and Blind Signal Separation (ICA2000)*, pages 251–256, Helsinki, Finland, 2000.

L. Xu. Least mean square error reconstruction principle for self-organizing neural-nets. *Neural Networks*, 6:627–648, 1993.

H.H. Yang, S. Amari, and A. Cichocki. Information-theoretic approach to blind separation of sources in nonlinear mixture. *Signal Processing*, 64(3):291–300, 1998.

K. Zhang and L. Chan. Kernel-based nonlinear independent component analysis. In *Proc. Int. Workshop on Independent Component Analysis and Signal Separation (ICA2007)*, pages 301–308, London, UK, Sept. 2007a.

K. Zhang and L. Chan. Nonlinear independent component analysis with minimum nonlinear distortion. In *the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 1127–1134, Corvallis, OR, US, Jun. 2007b.

K. Zhang and L. Chan. Extensions of ICA for causality discovery in the Hong Kong stock market. In *Proc. 13th International Conference on Neural Information Processing (ICONIP 2006)*, pages 400–409, Hong Kong, 2006.

Journal of Machine Learning Research 9 (2008) 2489-2490 Submitted 4/08; Revised 6/08; Published 11/08

On the Equivalence of Linear Dimensionality-Reducing Transformations

Marco Loog[*] M.LOOG@TUDELFT.NL
ICT Group
Delft University of Technology
Mekelweg 4
2628 CD Delft, The Netherlands

Editor: Leslie Pack Kaelbling

Abstract

In this JMLR volume, Ye (2008) demonstrates the essential equivalence of two sets of solutions to a generalized Fisher criterion used for linear dimensionality reduction (see Ye, 2005; Loog, 2007). Here, I point out the basic flaw in this new contribution.

Keywords: linear discriminant analysis, equivalence relation, linear subspaces, Bayes error

1. Introduction

Some time ago, Ye (2005) studied an optimization criterion for linear dimensionality reduction and tried to characterize the family of solutions to this objective function. The description, however, merely covers a part of the full solution set and is therefore, in fact, not at all a characterization. Loog (2007) has corrected this mistake, giving the proper, larger set of solutions. In this volume, Ye (2008) now demonstrates that the two solution sets are essentially equivalent.

In principle, Ye (2008) is correct and the two sets of dimension reducing transforms can indeed be considered equivalent. At the base of this fact is that mathematically speaking anything can be equivalent to anything else. The point I would like to convey, however, is that the equivalence considered is not essential and, as a result, the two sets are in fact essentially different. The main question in this is what is 'essential' in the context of supervised linear dimensionality reduction?

2. Essential Equivalence

Concerned with classification tasks, the performance of every dimensionality reduction criterion should primarily be discussed in relation to the Bayes error (see Fukunaga, 1990, Chapter 10). As such, transformations might be considered essentially equivalent if their Bayes errors in the reduces spaces are equal. A closely related definition is to consider transformations A and B equivalent if there is a nonsingular transformation T such that $A = T \circ B$ (see Fukunaga, 1990). The latter is more restrictive than the former as the existence of T implies an equal Bayes error for A and B, but the implication in the other direction does not necessarily hold. When A and B are linear and there is such a transform T, both of them span the same subspace of the original feature space, obviously

∗. Also in the Image Group, University of Copenhagen, Universitetsparken 1, 2100 Copenhagen Ø, Denmark.

resulting in the equality of the Bayes errors. Based on the foregoing, two linear transformations are also considered essentially equivalent if they span the same subspace.

Now, without providing any rationale, Ye (2008) declares two linear transformations A and B to be equivalent if there is a vector v such that $A(x_i - v) = B(x_i - v)$ for all feature vectors x_i in the training set. The following very simple examples demonstrate, however, why the latter definition of equivalence is flawed.

Let $x_1 = (0,0)^t$ and $x_2 = (1,0)^t$ be two training samples, $A = (1,0)$, $B = (-1,0)$, $C = (1,1)$, $D = (0,0)$, and $E = (0,1)$ be linear transformations, and let v equal to $(v_1, v_2)^t$. Now, firstly, one cannot have both $-v_1 = A(x_1 - v) = B(x_1 - v) = v_1$ and $1 - v_1 = A(x_2 - v) = B(x_2 - v) = -1 + v_1$, and therefore A is not equivalent to B even thought $A = -B$. That is, two transforms that trivially define the same subspace are apparently not equivalent. Secondly, $D(x_i - v) = 0 = E(x_i - v)$ shows that transforms spanning subspaces of different dimensions can be equivalent. Finally, a straightforward calculation shows that A and C are equivalent, that is, two transforms that obviously span different subspaces, and therefore most probably result in different Bayes errors, are considered equivalent.

3. In Conclusion

Maintaining that the equivalence relation in Ye (2008) is flawed, it directly follows that it cannot be concluded that the different sets of solutions as given by Loog (2007) and Ye (2005) are essentially equivalent. In fact, as should be obvious from Loog (2007), they are essentially different. Given that x_1 and x_2 (as defined above) come from two different classes, one can easily check that the solution set by Ye (2005) is given by $\{(a,0)|a \in \mathbb{R}\backslash 0\}$, that is, nondegenerate multiples of $A = (1,0)$, while the true set also contains transformations like $C = (1,1)$. Both define different subspaces and, generically, lead to different Bayes errors.

Acknowledgments

This research is supported by the Innovational Research Incentives Scheme of the Netherlands Research Organization [NWO], the Netherlands, and the Research Grant Program of the Faculty of Science, University of Copenhagen, Denmark.

References

K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1990.

M. Loog. A complete characterization of a family of solutions to a generalized fisher criterion. *Journal of Machine Learning Research*, 8:2121–2123, 2007.

J. Ye. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6:483–502, 2005.

J. Ye. Comments on the complete characterization of a family of solutions to a generalized fisher criterion. *Journal of Machine Learning Research*, 9:517–519, 2008.

Journal of Machine Learning Research 9 (2008) 2491-2521 Submitted 1/08; Revised 8/08; Published 11/08

SimpleMKL

Alain Rakotomamonjy ALAIN.RAKOTOMAMONJY@INSA-ROUEN.FR
LITIS EA 4108
Université de Rouen
76800 Saint Etienne du Rouvray, France

Francis R. Bach FRANCIS.BACH@MINES.ORG
INRIA - WILLOW Project - Team
Laboratoire d'Informatique de l'Ecole Normale Supérieure(CNRS/ENS/INRIA UMR 8548)
45, Rue d'Ulm, 75230 Paris, France

Stéphane Canu STEPHANE.CANU@INSA-ROUEN.FR
LITIS EA 4108
INSA de Rouen
76801 Saint Etienne du Rouvray, France

Yves Grandvalet YVES.GRANDVALET@UTC.FR
Idiap Research Institute, Centre du Parc
1920 Martigny, Switzerland[*]

Editor: Nello Cristianini

Abstract

Multiple kernel learning (MKL) aims at simultaneously learning a kernel and the associated predictor in supervised learning settings. For the support vector machine, an efficient and general multiple kernel learning algorithm, based on semi-infinite linear programming, has been recently proposed. This approach has opened new perspectives since it makes MKL tractable for large-scale problems, by iteratively using existing support vector machine code. However, it turns out that this iterative algorithm needs numerous iterations for converging towards a reasonable solution. In this paper, we address the MKL problem through a weighted 2-norm regularization formulation with an additional constraint on the weights that encourages sparse kernel combinations. Apart from learning the combination, we solve a standard SVM optimization problem, where the kernel is defined as a linear combination of multiple kernels. We propose an algorithm, named SimpleMKL, for solving this MKL problem and provide a new insight on MKL algorithms based on mixed-norm regularization by showing that the two approaches are equivalent. We show how SimpleMKL can be applied beyond binary classification, for problems like regression, clustering (one-class classification) or multiclass classification. Experimental results show that the proposed algorithm converges rapidly and that its efficiency compares favorably to other MKL algorithms. Finally, we illustrate the usefulness of MKL for some regressors based on wavelet kernels and on some model selection problems related to multiclass classification problems.

Keywords: multiple kernel learning, support vector machines, support vector regression, multiclass SVM, gradient descent

∗. Also at Heudiasyc, CNRS/Université de Technologie de Compiègne (UMR 6599), 60205 Compiègne, France.

1. Introduction

During the last few years, kernel methods, such as support vector machines (SVM) have proved to be efficient tools for solving learning problems like classification or regression (Schölkopf and Smola, 2001). For such tasks, the performance of the learning algorithm strongly depends on the data representation. In kernel methods, the data representation is implicitly chosen through the so-called kernel $K(x,x')$. This kernel actually plays two roles: it defines the similarity between two examples x and x', while defining an appropriate regularization term for the learning problem.

Let $\{x_i,y_i\}_{i=1}^{\ell}$ be the learning set, where x_i belongs to some input space $\mathcal{X}$ and y_i is the target value for pattern x_i. For kernel algorithms, the solution of the learning problem is of the form

$$f(x) = \sum_{i=1}^{\ell} \alpha_i^\star K(x,x_i) + b^\star, \tag{1}$$

where $\alpha_i^\star$ and $b^\star$ are some coefficients to be learned from examples, while $K(\cdot,\cdot)$ is a given positive definite kernel associated with a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$.

In some situations, a machine learning practitioner may be interested in more flexible models. Recent applications have shown that using multiple kernels instead of a single one can enhance the interpretability of the decision function and improve performances (Lanckriet et al., 2004a). In such cases, a convenient approach is to consider that the kernel $K(x,x')$ is actually a convex combination of *basis* kernels:

$$K(x,x') = \sum_{m=1}^{M} d_m K_m(x,x') \ , \quad \text{with } d_m \geq 0 \ , \ \sum_{m=1}^{M} d_m = 1 \ ,$$

where M is the total number of kernels. Each basis kernel K_m may either use the full set of variables describing x or subsets of variables stemming from different data sources (Lanckriet et al., 2004a). Alternatively, the kernels K_m can simply be classical kernels (such as Gaussian kernels) with different parameters. Within this framework, the problem of data representation through the kernel is then transferred to the choice of weights d_m.

Learning both the coefficients α_i and the weights d_m in a single optimization problem is known as the multiple kernel learning (MKL) problem. For binary classification, the MKL problem has been introduced by Lanckriet et al. (2004b), resulting in a quadratically constrained quadratic programming problem that becomes rapidly intractable as the number of learning examples or kernels become large.

What makes this problem difficult is that it is actually a convex but non-smooth minimization problem. Indeed, Bach et al. (2004a) have shown that the MKL formulation of Lanckriet et al. (2004b) is actually the dual of a SVM problem in which the weight vector has been regularized according to a mixed (ℓ_2,ℓ_1)-norm instead of the classical squared ℓ_2-norm. Bach et al. (2004a) have considered a smoothed version of the problem for which they proposed a SMO-like algorithm that enables to tackle medium-scale problems.

Sonnenburg et al. (2006) reformulate the MKL problem of Lanckriet et al. (2004b) as a semi-infinite linear program (SILP). The advantage of the latter formulation is that the algorithm addresses the problem by iteratively solving a classical SVM problem with a single kernel, for which many efficient toolboxes exist (Vishwanathan et al., 2003; Loosli et al., 2005; Chang and Lin, 2001), and a linear program whose number of constraints increases along with iterations. A very nice feature of this algorithm is that is can be extended to a large class of convex loss functions. For instance, Zien and Ong (2007) have proposed a multiclass MKL algorithm based on similar ideas.

In this paper, we present another formulation of the multiple learning problem. We first depart from the primal formulation proposed by Bach et al. (2004a) and further used by Bach et al. (2004b) and Sonnenburg et al. (2006). Indeed, we replace the mixed-norm regularization by a weighted ℓ_2-norm regularization, where the sparsity of the linear combination of kernels is controlled by a ℓ_1-norm constraint on the kernel weights. This new formulation of MKL leads to a smooth and convex optimization problem. By using a variational formulation of the mixed-norm regularization, we show that our formulation is equivalent to the ones of Lanckriet et al. (2004b), Bach et al. (2004a) and Sonnenburg et al. (2006).

The main contribution of this paper is to propose an efficient algorithm, named SimpleMKL, for solving the MKL problem, through a primal formulation involving a weighted ℓ_2-norm regularization. Indeed, our algorithm is simple, essentially based on a gradient descent on the SVM objective value. We iteratively determine the combination of kernels by a gradient descent wrapping a standard SVM solver, which is SimpleSVM in our case. Our scheme is similar to the one of Sonnenburg et al. (2006), and both algorithms minimize the same objective function. However, they differ in that we use reduced gradient descent in the primal, whereas Sonnenburg et al.'s SILP relies on cutting planes. We will empirically show that our optimization strategy is more efficient, with new evidences confirming the preliminary results reported in Rakotomamonjy et al. (2007).

Then, extensions of SimpleMKL to other supervised learning problems such as regression SVM, one-class SVM or multiclass SVM problems based on pairwise coupling are proposed. Although it is not the main purpose of the paper, we will also discuss the applicability of our approach to general convex loss functions.

This paper also presents several illustrations of the usefulness of our algorithm. For instance, in addition to the empirical efficiency comparison, we also show, in a SVM regression problem involving wavelet kernels, that automatic learning of the kernels leads to far better performances. Then we depict how our MKL algorithm behaves on some multiclass problems.

The paper is organized as follows. Section 2 presents the functional settings of our MKL problem and its formulation. Details on the algorithm and discussion of convergence and computational complexity are given in Section 3. Extensions of our algorithm to other SVM problems are discussed in Section 4 while experimental results dealing with computational complexity or with comparison with other model selection methods are presented in Section 5.

A SimpleMKL toolbox based on Matlab code is available at `http://www.mloss.org`. This toolbox is an extension of our SVM-KM toolbox (Canu et al., 2003).

2. Multiple Kernel Learning Framework

In this section, we present our MKL formulation and derive its dual. In the sequel, i and j are indices on examples, whereas m is the kernel index. In order to lighten notations, we omit to specify that summations on i and j go from 1 to ℓ, and that summations on m go from 1 to M.

2.1 Functional Framework

Before entering into the details of the MKL optimization problem, we first present the functional framework adopted for multiple kernel learning. Assume $K_m, m = 1,...,M$ are M positive definite kernels on the same input space $\mathcal{X}$, each of them being associated with an RKHS $\mathcal{H}_m$ endowed with an inner product $\langle \cdot, \cdot \rangle_m$. For any m, let d_m be a non-negative coefficient and $\mathcal{H}'_m$ be the Hilbert space

derived from $\mathcal{H}_m$ as follows:

$$\mathcal{H}'_m = \{ f \mid f \in \mathcal{H}_m : \frac{\|f\|_{\mathcal{H}_m}}{d_m} < \infty \} \ ,$$

endowed with the inner product

$$\langle f, g \rangle_{\mathcal{H}'_m} = \frac{1}{d_m} \langle f, g \rangle_m \ .$$

In this paper, we use the convention that $\frac{x}{0} = 0$ if $x = 0$ and ∞ otherwise. This means that, if $d_m = 0$ then a function f belongs to the Hilbert space $\mathcal{H}'_m$ only if $f = 0 \in \mathcal{H}_m$. In such a case, $\mathcal{H}'_m$ is restricted to the null element of $\mathcal{H}_m$.

Within this framework, $\mathcal{H}'_m$ is a RKHS with kernel $K(x,x') = d_m \, K_m(x,x')$ since

$$\begin{aligned} \forall f \in \mathcal{H}'_m \subseteq \mathcal{H}_m \ , \quad f(x) &= \langle f(\cdot), K_m(x,\cdot) \rangle_m \\ &= \frac{1}{d_m} \langle f(\cdot), d_m K_m(x,\cdot) \rangle_m \\ &= \langle f(\cdot), d_m K_m(x,\cdot) \rangle_{\mathcal{H}'_m} \ . \end{aligned}$$

Now, if we define $\mathcal{H}$ as the direct sum of the spaces $\mathcal{H}'_m$, that is,

$$\mathcal{H} = \bigoplus_{m=1}^{M} \mathcal{H}'_m \ ,$$

then, a classical result on RKHS (Aronszajn, 1950) says that $\mathcal{H}$ is a RKHS of kernel

$$K(x,x') = \sum_{m=1}^{M} d_m K_m(x,x') \ .$$

Owing to this simple construction, we have built a RKHS $\mathcal{H}$ for which any function is a sum of functions belonging to $\mathcal{H}_m$. In our framework, MKL aims at determining the set of coefficients $\{d_m\}$ within the learning process of the decision function. The multiple kernel learning problem can thus be envisioned as learning a predictor belonging to an adaptive hypothesis space endowed with an adaptive inner product. The forthcoming sections explain how we solve this problem.

2.2 Multiple Kernel Learning Primal Problem

In the SVM methodology, the decision function is of the form given in Equation (1), where the optimal parameters $\alpha_i^\star$ and $b^\star$ are obtained by solving the dual of the following optimization problem:

$$\begin{array}{ll} \min\limits_{f,b,\xi} & \frac{1}{2}\|f\|^2_{\mathcal{H}} + C \sum\limits_i \xi_i \\ \text{s.t.} & y_i(f(x_i)+b) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \ . \end{array}$$

In the MKL framework, one looks for a decision function of the form $f(x) + b = \sum_m f_m(x) + b$, where each function f_m belongs to a different RKHS $\mathcal{H}_m$ associated with a kernel K_m. According to the above functional framework and inspired by the multiple smoothing splines framework of

Wahba (1990, chap. 10), we propose to address the MKL SVM problem by solving the following convex problem (see proof in appendix), which we will be referred to as the primal MKL problem:

$$\begin{array}{ll} \min\limits_{\{f_m\},b,\xi,d} & \frac{1}{2}\sum\limits_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \\ \text{s.t.} & y_i\sum\limits_m f_m(x_i) + y_i b \geq 1-\xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \\ & \sum\limits_m d_m = 1\ , \quad d_m \geq 0 \quad \forall m\ , \end{array} \tag{2}$$

where each d_m controls the squared norm of f_m in the objective function.

The smaller d_m is, the smoother f_m (as measured by $\|f_m\|_{\mathcal{H}_m}$) should be. When $d_m = 0$, $\|f_m\|_{\mathcal{H}_m}$ has also to be equal to zero to yield a finite objective value. The ℓ_1-norm constraint on the vector d is a sparsity constraint that will force some d_m to be zero, thus encouraging sparse basis kernel expansions.

2.3 Connections With mixed-norm Regularization Formulation of MKL

The MKL formulation introduced by Bach et al. (2004a) and further developed by Sonnenburg et al. (2006) consists in solving an optimization problem expressed in a functional form as

$$\begin{array}{ll} \min\limits_{\{f\},b,\xi} & \frac{1}{2}\left(\sum\limits_m \|f_m\|_{\mathcal{H}_m}\right)^2 + C\sum\limits_i \xi_i \\ \text{s.t.} & y_i\sum\limits_m f_m(x_i) + y_i b \geq 1-\xi_i \ \ \forall i \\ & \xi_i \geq 0 \ \ \forall i. \end{array} \tag{3}$$

Note that the objective function of this problem is not smooth since $\|f_m\|_{\mathcal{H}_m}$ is not differentiable at $f_m = 0$. However, what makes this formulation interesting is that the mixed-norm penalization of $f = \sum_m f_m$ is a soft-thresholding penalizer that leads to a sparse solution, for which the algorithm performs kernel selection (Bach, 2008). We have stated in the previous section that our problem should also lead to sparse solutions. In the following, we show that the formulations (2) and (3) are equivalent.

For this purpose, we simply show that the variational formulation of the mixed-norm regularization is equal to the weighted 2-norm regularization, (which is a particular case of a more general equivalence proposed by Micchelli and Pontil 2005), that is, by Cauchy-Schwartz inequality, for any vector d on the simplex:

$$\begin{aligned} \left(\sum_m \|f_m\|_{\mathcal{H}_m}\right)^2 &= \left(\sum_m \frac{\|f_m\|_{\mathcal{H}_m}}{d_m^{1/2}} d_m^{1/2}\right)^2 \\ &\leqslant \left(\sum_m \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m}\right)\left(\sum_m d_m\right) \\ &\leqslant \sum_m \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m}\ , \end{aligned}$$

where equality is met when $d_m^{1/2}$ is proportional to $\|f_m\|_{\mathcal{H}_m}/d_m^{1/2}$, that is:

$$d_m = \frac{\|f_m\|_{\mathcal{H}_m}}{\sum_q \|f_q\|_{\mathcal{H}_q}} \ ,$$

which leads to

$$\min_{d_m \geq 0, \sum_m d_m = 1} \sum_m \frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m} = \left(\sum_m \|f_m\|_{\mathcal{H}_m} \right)^2 \ .$$

Hence, owing to this variational formulation, the non-smooth mixed-norm objective function of problem (3) has been turned into a smooth objective function in problem (2). Although the number of variables has increased, we will see that this problem can be solved more efficiently.

2.4 The MKL Dual Problem

The dual problem is a key point for deriving MKL algorithms and for studying their convergence properties. Since our primal problem (2) is equivalent to the one of Bach et al. (2004a), they lead to the same dual. However, our primal formulation being convex and differentiable, it provides a simple derivation of the dual, that does not use conic duality.

The Lagrangian of problem (2) is

$$\begin{aligned} \mathcal{L} \ = \ & \frac{1}{2}\sum_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum_i \xi_i + \sum_i \alpha_i \left(1 - \xi_i - y_i \sum_m f_m(x_i) - y_i b\right) - \sum_i \nu_i \xi_i \\ & + \lambda\left(\sum_m d_m - 1\right) - \sum_m \eta_m d_m \ , \end{aligned} \tag{4}$$

where α_i and ν_i are the Lagrange multipliers of the constraints related to the usual SVM problem, whereas λ and η_m are associated to the constraints on d_m. When setting to zero the gradient of the Lagrangian with respect to the primal variables, we get the following

$$\begin{aligned} &\text{(a)} \quad \frac{1}{d_m} f_m(\cdot) = \sum_i \alpha_i y_i K_m(\cdot, x_i) \ , \quad \forall m, \\ &\text{(b)} \quad \sum_i \alpha_i y_i = 0, \\ &\text{(c)} \quad C - \alpha_i - \nu_i = 0 \ , \quad \forall i, \\ &\text{(d)} \quad -\frac{1}{2}\frac{\|f_m\|^2_{\mathcal{H}_m}}{d_m^2} + \lambda - \eta_m = 0 \ , \quad \forall m \ . \end{aligned} \tag{5}$$

We note again here that $f_m(\cdot)$ has to go to 0 if the coefficient d_m vanishes. Plugging these optimality conditions in the Lagrangian gives the dual problem

$$\begin{aligned} \max_{\alpha_i, \lambda} \quad & \sum_i \alpha_i - \lambda \\ \text{s.t.} \quad & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \quad \forall i \\ & \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K_m(x_i, x_j) \leq \lambda \ , \quad \forall m \ . \end{aligned} \tag{6}$$

This dual problem[1] is difficult to optimize due to the last constraint. This constraint may be moved to the objective function, but then, the latter becomes non-differentiable causing new difficulties (Bach et al., 2004a). Hence, in the forthcoming section, we propose an approach based on the minimization of the primal. In this framework, we benefit from differentiability which allows for an efficient derivation of an approximate primal solution, whose accuracy will be monitored by the duality gap.

3. Algorithm for Solving the MKL Primal Problem

One possible approach for solving problem (2) is to use the alternate optimization algorithm applied by Grandvalet and Canu (1999, 2003) in another context. In the first step, problem (2) is optimized with respect to f_m, b and ξ, with d fixed. Then, in the second step, the weight vector d is updated to decrease the objective function of problem (2), with f_m, b and ξ being fixed. In Section 2.3, we showed that the second step can be carried out in closed form. However, this approach lacks convergence guarantees and may lead to numerical problems, in particular when some elements of d approach zero (Grandvalet, 1998). Note that these numerical problems can be handled by introducing a perturbed version of the alternate algorithm as shown by Argyriou et al. (2008).

Instead of using an alternate optimization algorithm, we prefer to consider here the following constrained optimization problem:

$$\min_d J(d) \quad \text{such that} \quad \sum_{m=1}^{M} d_m = 1,\ d_m \geq 0 \ , \tag{7}$$

where

$$J(d) = \left\{ \begin{array}{ll} \min\limits_{\{f\},b,\xi} & \frac{1}{2}\sum\limits_m \frac{1}{d_m}\|f_m\|^2_{\mathcal{H}_m} + C\sum\limits_i \xi_i \quad \forall i \\ \text{s.t.} & y_i \sum\limits_m f_m(x_i) + y_i b \geq 1 - \xi_i \\ & \xi_i \geq 0 \quad \forall i \ . \end{array} \right. \tag{8}$$

We show below how to solve problem (7) on the simplex by a simple gradient method. We will first note that the objective function $J(d)$ is actually an optimal SVM objective value. We will then discuss the existence and computation of the gradient of $J(\cdot)$, which is at the core of the proposed approach.

3.1 Computing the Optimal SVM Value and its Derivatives

The Lagrangian of problem (8) is identical to the first line of Equation (4). By setting to zero the derivatives of this Lagrangian according to the primal variables, we get conditions (5) (a) to (c), from which we derive the associated dual problem

$$\begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum\limits_{i,j}\alpha_i\alpha_j y_i y_j \sum\limits_m d_m K_m(x_i,x_j) + \sum\limits_i \alpha_i \\ \text{with} & \sum\limits_i \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \quad \forall i \ , \end{array} \tag{9}$$

1. Note that Bach et al. (2004a) formulation differs slightly, in that the kernels are weighted by some pre-defined coefficients that were not considered here.

which is identified as the standard SVM dual formulation using the combined kernel $K(x_i,x_j) = \sum_m d_m K_m(x_i,x_j)$. Function $J(d)$ is defined as the optimal objective value of problem (8). Because of strong duality, $J(d)$ is also the objective value of the dual problem:

$$J(d) = -\frac{1}{2}\sum_{i,j}\alpha_i^\star\alpha_j^\star y_i y_j \sum_m d_m K_m(x_i,x_j) + \sum_i \alpha_i^\star \quad , \tag{10}$$

where $\alpha^\star$ maximizes (9). Note that the objective value $J(d)$ can be obtained by any SVM algorithm. Our method can thus take advantage of any progress in single kernel algorithms. In particular, if the SVM algorithm we use is able to handle large-scale problems, so will our MKL algorithm. Thus, the overall complexity of SimpleMKL is tied to the one of the single kernel SVM algorithm.

From now on, we assume that each Gram matrix $(K_m(x_i,x_j))_{i,j}$ is positive definite, with all eigenvalues greater than some $\eta > 0$ (to enforce this property, a small ridge may be added to the diagonal of the Gram matrices). This implies that, for any admissible value of d, the dual problem is strictly concave with convexity parameter η (Lemaréchal and Sagastizabal, 1997). In turn, this strict concavity property ensures that $\alpha^\star$ is unique, a characteristic that eases the analysis of the differentiability of $J(\cdot)$.

Existence and computation of derivatives of optimal value functions such as $J(\cdot)$ have been largely discussed in the literature. For our purpose, the appropriate reference is Theorem 4.1 in Bonnans and Shapiro (1998), which has already been applied by Chapelle et al. (2002) for tuning squared-hinge loss SVM. This theorem is reproduced in the appendix for self-containedness. In a nutshell, it says that differentiability of $J(d)$ is ensured by the uniqueness of $\alpha^\star$, and by the differentiability of the objective function that gives $J(d)$. Furthermore, the derivatives of $J(d)$ can be computed as if $\alpha^\star$ were not to depend on d. Thus, by simple differentiation of the dual function (9) with respect to d_m, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{i,j}\alpha_i^\star\alpha_j^\star y_i y_j K_m(x_i,x_j) \qquad \forall m \ . \tag{11}$$

We will see in the sequel that the applicability of this theorem can be extended to other SVM problems. Note that complexity of the gradient computation is of the order of $m \cdot n_{SV}^2$, with n_{SV} being the number of support vectors for the current d.

3.2 Reduced Gradient Algorithm

The optimization problem we have to deal with in (7) is a non-linear objective function with constraints over the simplex. With our positivity assumption on the kernel matrices, $J(\cdot)$ is convex and differentiable with Lipschitz gradient (Lemaréchal and Sagastizabal, 1997). The approach we use for solving this problem is a reduced gradient method, which converges for such functions (Luenberger, 1984).

Once the gradient of $J(d)$ is computed, d is updated by using a descent direction ensuring that the equality constraint and the non-negativity constraints on d are satisfied. We handle the equality constraint by computing the reduced gradient (Luenberger, 1984, Chap. 11). Let d_μ be a non-zero entry of d, the reduced gradient of $J(d)$, denoted $\nabla_{red}J$, has components:

$$[\nabla_{red}J]_m = \frac{\partial J}{\partial d_m} - \frac{\partial J}{\partial d_\mu} \ \ \forall m \neq \mu \ , \text{ and } [\nabla_{red}J]_\mu = \sum_{m\neq\mu}\left(\frac{\partial J}{\partial d_\mu} - \frac{\partial J}{\partial d_m}\right) \ .$$

We chose μ to be the index of the largest component of vector d, for better numerical stability (Bonnans, 2006).

The positivity constraints have also to be taken into account in the descent direction. Since we want to minimize $J(\cdot)$, $-\nabla_{red}J$ is a descent direction. However, if there is an index m such that $d_m = 0$ and $[\nabla_{red}J]_m > 0$, using this direction would violate the positivity constraint for d_m. Hence, the descent direction for that component is set to 0. This gives the descent direction for updating d as

$$D_m = \begin{cases} 0 & \text{if } d_m = 0 \text{ and } \frac{\partial J}{\partial d_m} - \frac{\partial J}{\partial d_\mu} > 0 \\ -\frac{\partial J}{\partial d_m} + \frac{\partial J}{\partial d_\mu} & \text{if } d_m > 0 \text{ and } m \neq \mu \\ \sum_{g \neq \mu, d_\nu > 0} \left(\frac{\partial J}{\partial d_\nu} - \frac{\partial J}{\partial d_\mu} \right) & \text{for } m = \mu \ . \end{cases} \tag{12}$$

The usual updating scheme is $d \leftarrow d + \gamma D$, where γ is the step size. Here, as detailed in Algorithm 1, we go one step beyond: once a descent direction D has been computed, we first look for the maximal admissible step size in that direction and check whether the objective value decreases or not. The maximal admissible step size corresponds to a component, say d_ν, set to zero. If the objective value decreases, d is updated, we set $D_\nu = 0$ and normalize D to comply with the equality constraint. This procedure is repeated until the objective value stops decreasing. At this point, we look for the optimal step size γ, which is determined by using a one-dimensional line search, with proper stopping criterion, such as Armijo's rule, to ensure global convergence.

In this algorithm, computing the descent direction and the line search are based on the evaluation of the objective function $J(\cdot)$, which requires solving an SVM problem. This may seem very costly but, for small variations of d, learning is very fast when the SVM solver is initialized with the previous values of $\alpha^\star$ (DeCoste and Wagstaff., 2000). Note that the gradient of the cost function is not computed after each update of the weight vector d. Instead, we take advantage of an easily updated descent direction as long as the objective value decreases. We will see in the numerical experiments that this approach saves a substantial amount of computation time compared to the usual update scheme where the descent direction is recomputed after each update of d. Note that we have also investigated gradient projection algorithms (Bertsekas, 1999, Chap 2.3), but this turned out to be slightly less efficient than the proposed approach, and we will not report these results.

The algorithm is terminated when a stopping criterion is met. This stopping criterion can be either based on the duality gap, the KKT conditions, the variation of d between two consecutive steps or, even more simply, on a maximal number of iterations. Our implementation, based on the duality gap, is detailed in the forthcoming section.

3.3 Optimality Conditions

In a convex constrained optimization algorithm such as the one we are considering, we have the opportunity to check for proper optimality conditions such as the KKT conditions or the duality gap (the difference between primal and dual objective values), which should be zero at the optimum. From the primal and dual objectives provided respectively in (2) and (6), the MKL duality gap is

$$\text{DualGap} = J(d^\star) - \sum_i \alpha_i^\star + \frac{1}{2} \max_m \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) \ ,$$

Algorithm 1 SimpleMKL algorithm

set $d_m = \frac{1}{M}$ for $m = 1, \ldots, M$
while stopping criterion not met **do**
  compute $J(d)$ by using an SVM solver with $K = \sum_m d_m K_m$
  compute $\frac{\partial J}{\partial d_m}$ for $m = 1, \ldots, M$ and descent direction D (12).
  set $\mu = \underset{m}{\operatorname{argmax}}\, d_m$, $J^\dagger = 0$, $d^\dagger = d$, $D^\dagger = D$
  while $J^\dagger < J(d)$ **do** {descent direction update}
    $d = d^\dagger$, $D = D^\dagger$
    $\nu = \underset{\{m|D_m<0\}}{\operatorname{argmin}} -d_m/D_m$, $\gamma_{\max} = -d_\nu/D_\nu$
    $d^\dagger = d + \gamma_{\max} D$, $D^\dagger_\mu = D_\mu - D_\nu$, $D^\dagger_\nu = 0$
    compute $J^\dagger$ by using an SVM solver with $K = \sum_m d^\dagger_m K_m$
  end while
  line search along D for $\gamma \in [0, \gamma_{\max}]$ {calls an SVM solver for each γ trial value}
  $d \leftarrow d + \gamma D$
end while

where $d^\star$ and $\{\alpha_i^\star\}$ are optimal primal and dual variables, and $J(d^\star)$ depends implicitly on optimal primal variables $\{f_m^\star\}$, $b^\star$ and $\{\xi_i^\star\}$. If $J(d^\star)$ has been obtained through the dual problem (9), then this MKL duality gap can also be computed from the single kernel SVM algorithm duality gap DG_{SVM}. Indeed, Equation (10) holds only when the single kernel SVM algorithm returns an exact solution with $DG_{\text{SVM}} = 0$. Otherwise, we have

$$DG_{\text{SVM}} = J(d^\star) + \frac{1}{2}\sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_m d_m^\star K_m(x_i, x_j) - \sum_i \alpha_i^\star$$

then the MKL duality gap becomes

$$\text{DualGap} = DG_{\text{SVM}} - \frac{1}{2}\sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_m d_m^\star K_m(x_i, x_j) + \frac{1}{2}\max_m \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) \quad .$$

Hence, it can be obtained with a small additional computational cost compared to the SVM duality gap.

In iterative procedures, it is common to stop the algorithm when the optimality conditions are respected up to a tolerance threshold ε. Obviously, SimpleMKL has no impact on DG_{SVM}, hence, one may assume, as we did here, that DG_{SVM} needs not to be monitored. Consequently, we terminate the algorithm when

$$\max_m \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j K_m(x_i, x_j) - \sum_{i,j} \alpha_i^\star \alpha_j^\star y_i y_j \sum_m d_m^\star K_m(x_i, x_j) \leq \varepsilon \quad . \tag{13}$$

For some of the other MKL algorithms that will be presented in Section 4, the dual function may be more difficult to derive. Hence, it may be easier to rely on approximate KKT conditions as a stopping criterion. For the general MKL problem (7), the first order optimality conditions are

obtained through the KKT conditions:

$$\begin{aligned} \frac{\partial J}{\partial d_m} + \lambda - \eta_m &= 0 \quad \forall m \ , \\ \eta_m \cdot d_m &= 0 \quad \forall m \ , \end{aligned}$$

where λ and $\{\eta_m\}$ are respectively the Lagrange multipliers for the equality and inequality constraints of (7). These KKT conditions imply

$$\begin{aligned} \frac{\partial J}{\partial d_m} &= -\lambda \quad \text{if } d_m > 0 \ , \\ \frac{\partial J}{\partial d_m} &\geq -\lambda \quad \text{if } d_m = 0 \ . \end{aligned}$$

However, as Algorithm 1 is not based on the Lagrangian formulation of problem (7), λ is not computed. Hence, we derive approximate necessary optimality conditions to be used for termination criterion. Let's define $dJ_{\min}$ and $dJ_{\max}$ as

$$dJ_{\min} = \min_{\{d_m | d_m > 0\}} \frac{\partial J}{\partial d_m} \quad \text{and} \quad dJ_{\max} = \max_{\{d_m | d_m > 0\}} \frac{\partial J}{\partial d_m} \ ,$$

then, the necessary optimality conditions are approximated by the following termination conditions:

$$|dJ_{\min} - dJ_{\max}| \leq \varepsilon \quad \text{and} \quad \frac{\partial J}{\partial d_m} \geq dJ_{\max} \quad \text{if } d_m = 0 \ .$$

In other words, we are considered at the optimum when the gradient components for all positive d_m lie in a ε-tube and when all gradient components for vanishing d_m are outside this tube. Note that these approximate necessary optimality conditions are available right away for any differentiable objective function $J(d)$.

3.4 Cutting Planes, Steepest Descent and Computational Complexity

As we stated in the introduction, several algorithms have been proposed for solving the original MKL problem defined by Lanckriet et al. (2004b). All these algorithms are based on equivalent formulations of the same dual problem; they all aim at providing a pair of optimal vectors (d, α).

In this subsection, we contrast SimpleMKL with its closest relative, the SILP algorithm of Sonnenburg et al. (2005, 2006). Indeed, from an implementation point of view, the two algorithms are alike, since they are wrapping a standard single kernel SVM algorithm. This feature makes both algorithms very easy to implement. They, however, differ in computational efficiency, because the kernel weights d_m are optimized in quite different ways, as detailed below.

Let us first recall that our differentiable function $J(d)$ is defined as:

$$J(d) = \left\{ \begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2} \sum\limits_{i,j} \alpha_i \alpha_j y_i y_j \sum\limits_m d_m K_m(x_i, x_j) + \sum\limits_i \alpha_i \\ \text{with} & \sum\limits_i \alpha_i y_i = 0, \qquad C \geq \alpha_i \geq 0 \quad \forall i \ , \end{array} \right.$$

and both algorithms aim at minimizing this differentiable function. However, using a SILP approach in this case, does not take advantage of the smoothness of the objective function.

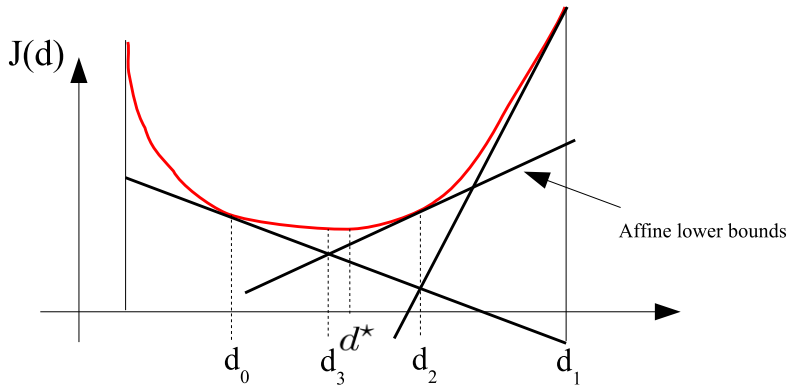


Figure 1: Illustrating three iterations of the SILP algorithm and a gradient descent algorithm for a one-dimensional problem. This dimensionality is not representative of the MKL framework, but our aim is to illustrate the typical oscillations of cutting planes around the optimal solution (with iterates d_0 to d_3). Note that computing an affine lower bound at a given d requires a gradient computation. Provided the step size is chosen correctly, gradient descent converges directly towards the optimal solution without overshooting (from d_0 to $d^\star$).

The SILP algorithm of Sonnenburg et al. (2006) is a *cutting plane* method to minimize J with respect to d. For each value of d, the best α is found and leads to an affine lower bound on $J(d)$. The number of lower bounding affine functions increases as more (d,α) pairs are computed, and the next candidate vector d is the minimizer of the current lower bound on $J(d)$, that is, the maximum over all the affine functions. Cutting planes method do converge but they are known for their instability, notably when the number of lower-bounding affine functions is small: the approximation of the objective function is then loose and the iterates may oscillate (Bonnans et al., 2003). Our steepest descent approach, with the proposed line search, does not suffer from instability since we have a differentiable function to minimize. Figure 1 illustrates the behaviour of both algorithms in a simple case, with oscillations for cutting planes and direct convergence for gradient descent.

Section 5 evaluates how these oscillations impact on the computational time of the SILP algorithm on several examples. These experiments show that our algorithm needs less costly gradient computations. Conversely, the line search in the gradient base approach requires more SVM retrainings in the process of querying the objective function. However, the computation time per SVM training is considerably reduced, since the gradient based approach produces estimates of d on a smooth trajectory, so that the previous SVM solution provides a good guess for the current SVM training. In SILP, with the oscillatory subsequent approximations of $d^\star$, the benefit of warm-start training severely decreases.

3.5 Convergence Analysis

In this paragraph, we briefly discuss the convergence of the algorithm we propose. We first suppose that problem (8) is always exactly solved, which means that the duality gap of such problem is 0. With such conditions, the gradient computation in (11) is exact and thus our algorithm performs reduced gradient descent on a continuously differentiable function $J(\cdot)$ (remember that we have

assumed that the kernel matrices are positive definite) defined on the simplex $\{d | \sum_m d_m = 1, d_m \geq 0\}$, which does converge to the global minimum of J (Luenberger, 1984).

However, in practice, problem (8) is not solved exactly since most SVM algorithms will stop when the duality gap is smaller than a given ε. In this case, the convergence of our reduced gradient method is no more guaranteed by standard arguments. Indeed, the output of the approximately solved SVM leads only to an ε-subgradient (Bonnans et al., 2003; Bach et al., 2004a). This situation is more difficult to analyze and we plan to address it thoroughly in future work (see for instance D'Aspremont 2008 for an example of such analysis in a similar context).

4. Extensions

In this section, we discuss how the proposed algorithm can be simply extended to other SVM algorithms such as SVM regression, one-class SVM or pairwise multiclass SVM algorithms. More generally, we will discuss other loss functions that can be used within our MKL algorithms.

4.1 Extensions to Other SVM Algorithms

The algorithm we described in the previous section focuses on binary classification SVMs, but it is worth noting that our MKL algorithm can be extended to other SVM algorithms with only little changes. For SVM regression with the ε-insensitive loss, or clustering with the one-class soft margin loss, the problem only changes in the definition of the objective function $J(d)$ in (8).

For SVM regression (Vapnik et al., 1997; Schölkopf and Smola, 2001), we have

$$J(d) = \left\{ \begin{array}{ll} \min\limits_{f_m, b, \xi_i} & \frac{1}{2} \sum\limits_m \frac{1}{d_m} \| f_m \|^2_{\mathcal{H}_m} + C \sum\limits_i (\xi_i + \xi_i^*) \\ \text{s.t.} & y_i - \sum\limits_m f_m(x_i) - b \leq \varepsilon + \xi_i \quad \forall i \\ & \sum\limits_m f_m(x_i) + b - y_i \leq \varepsilon + \xi_i^* \quad \forall i \\ & \xi_i \geq 0, \xi_i^* \leq 0 \quad \forall i \quad , \end{array} \right. \tag{14}$$

and for one-class SVMs (Schölkopf and Smola, 2001), we have:

$$J(d) = \left\{ \begin{array}{ll} \min\limits_{f_m, b, \xi_i} & \frac{1}{2} \sum\limits_m \frac{1}{d_m} \| f_m \|^2_{\mathcal{H}_m} + \frac{1}{\nu \ell} \sum\limits_i \xi_i - b \\ \text{s.t.} & \sum\limits_m f_m(x_i) \geq b - \xi_i \\ & \xi_i \geq 0 \quad . \end{array} \right.$$

Again, $J(d)$ can be defined according to the dual functions of these two optimization problems, which are respectively

$$J(d) = \left\{ \begin{array}{ll} \max\limits_{\alpha, \beta} & \sum\limits_i (\beta_i - \alpha_i) y_i - \varepsilon \sum\limits_i (\beta_i + \alpha_i) - \frac{1}{2} \sum\limits_{i,j} (\beta_i - \alpha_i)(\beta_j - \alpha_j) \sum\limits_m d_m K_m(x_i, x_j) \\ \text{with} & \sum\limits_i (\beta_i - \alpha_i) = 0 \\ & 0 \leq \alpha_i \ , \ \beta_i \leq C, \quad \forall i \ , \end{array} \right.$$

and

$$J(d) = \left\{ \begin{array}{ll} \max\limits_{\alpha} & -\frac{1}{2}\sum\limits_{i,j}\alpha_i\alpha_j\sum\limits_m d_m K_m(x_i,x_j) \\ \text{with} & 0 \le \alpha_i \le \frac{1}{\nu\ell} \quad \forall i \\ & \sum\limits_i \alpha_i = 1 \ , \end{array} \right.$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are Lagrange multipliers.

Then, as long as $J(d)$ is differentiable, a property strictly related to the strict concavity of its dual function, our descent algorithm can still be applied. The main effort for the extension of our algorithm is the evaluation of $J(d)$ and the computation of its derivatives. Like for binary classification SVM, $J(d)$ can be computed by means of efficient off-the-shelf SVM solvers and the gradient of $J(d)$ is easily obtained through the dual problems. For SVM regression, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{i,j}(\beta_i^\star - \alpha_i^\star)(\beta_j^\star - \alpha_j^\star)K_m(x_i,x_j) \qquad \forall m \ ,$$

and for one-class SVM, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2}\sum_{i,j}\alpha_i^\star\alpha_j^\star K_m(x_i,x_j) \qquad \forall m \ ,$$

where $\alpha_i^\star$ and $\beta_i^\star$ are the optimal values of the Lagrange multipliers. These examples illustrate that extending SimpleMKL to other SVM problems is rather straightforward. This observation is valid for other SVM algorithms (based for instance on the ν parameter, a squared hinge loss or squared-ε tube) that we do not detail here. Again, our algorithm can be used provided $J(d)$ is differentiable, by plugging in the algorithm the function that evaluates the objective value $J(d)$ and its gradient. Of course, the duality gap may be considered as a stopping criterion if it can be computed.

4.2 Multiclass Multiple Kernel Learning

With SVMs, multiclass problems are customarily solved by combining several binary classifiers. The well-known *one-against-all* and *one-against-one* approaches are the two most common ways for building a multiclass decision function based on pairwise decision functions. Multiclass SVM may also be defined right away as the solution of a global optimization problem (Weston and Watkins, 1999; Crammer and Singer, 2001), that may also be addressed with structured-output SVM (Tsochantaridis et al., 2005). Very recently, an MKL algorithm based on structured-output SVM has been proposed by Zien and Ong (2007). This work extends the work of Sonnenburg et al. (2006) to multiclass problems, with an MKL implementation still based on a QCQP or SILP approach.

Several works have compared the performance of multiclass SVM algorithms (Duan and Keerthi, 2005; Hsu and Lin, 2002; Rifkin and Klautau, 2004). In this subsection, we do not deal with this aspect; we explain how SimpleMKL can be extended to pairwise SVM multiclass implementations. The problem of applying our algorithm to structured-output SVM will be briefly discussed later.

Suppose we have a multiclass problem with P classes. For a *one-against-all* multiclass SVM, we need to train P binary SVM classifiers, where the p-th classifier is trained by considering all examples of class p as positive examples while all other examples are considered negative. For a *one-against-one* multiclass problem, $P(P-1)/2$ binary SVM classifiers are built from all pairs

of distinct classes. Our multiclass MKL extension of SimpleMKL differs from the binary version only in the definition of a new cost function $J(d)$. As we now look for the combination of kernels that *jointly* optimizes all the pairwise decision functions, the objective function we want to optimize according to the kernel weights $\{d_m\}$ is:

$$J(d) = \sum_{p \in \mathcal{P}} J_p(d) \ ,$$

where $\mathcal{P}$ is the set of all pairs to be considered, and $J_p(d)$ is the binary SVM objective value for the classification problem pertaining to pair p.

Once the new objective function is defined, the lines of Algorithm 1 still apply. The gradient of $J(d)$ is still very simple to obtain, since owing to linearity, we have:

$$\frac{\partial J}{\partial d_m} = -\frac{1}{2} \sum_{p \in \mathcal{P}} \sum_{i,j} \alpha^\star_{i,p} \alpha^\star_{j,p} y_i y_j K_m(x_i, x_j) \qquad \forall m \ ,$$

where $\alpha_{j,p}$ is the Lagrange multiplier of the j-th example involved in the p-th decision function. Note that those Lagrange multipliers can be obtained *independently* for each pair.

The approach described above aims at finding the combination of kernels that *jointly* optimizes all binary classification problems: this one set of features should maximize the sum of margins. Another possible and straightforward approach consists in running independently SimpleMKL for each classification task. However, this choice is likely to result in as many combinations of kernels as there are binary classifiers.

4.3 Other Loss Functions

Multiple kernel learning has been of great interest and since the seminal work of Lanckriet et al. (2004b), several works on this topic have flourished. For instance, multiple kernel learning has been transposed to least-square fitting and logistic regression (Bach et al., 2004b). Independently, several authors have applied mixed-norm regularization, such as the additive spline regression model of Grandvalet and Canu (1999). This type of regularization, which is now known as the *group lasso*, may be seen as a linear version of multiple kernel learning (Bach, 2008). Several algorithms have been proposed for solving the group lasso problem. Some of them are based on projected gradient or on coordinate descent algorithm. However, they all consider the non-smooth version of the problem.

We previously mentioned that Zien and Ong (2007) have proposed an MKL algorithm based on structured-output SVMs. For such problem, the loss function, which differs from the usual SVM hinge loss, leads to an algorithm based on cutting planes instead of the usual QP approach.

Provided the gradient of the objective value can be obtained, our algorithm can be applied to group lasso and structured-output SVMs. The key point is whether the theorem of Bonnans et al. (2003) can be applied or not. Although we have not deeply investigated this point, we think that many problems comply with this requirement, but we leave these developments for future work.

4.4 Approximate Regularization Path

SimpleMKL requires the setting of the usual SVM hyperparameter C, which usually needs to be tuned for the problem at hand. For doing so, a practical and useful technique is to compute the so-called regularization path, which describes the set of solutions as C varies from 0 to ∞.

Exact path following techniques have been derived for some specific problems like SVMs or the lasso (Hastie et al., 2004; Efron et al., 2004). Besides, regularization paths can be sampled by predictor-corrector methods (Rosset, 2004; Bach et al., 2004b).

For model selection purposes, an approximation of the regularization path may be sufficient. This approach has been applied for instance by Koh et al. (2007) in regularized logistic regression.

Here, we compute an approximate regularization path based on a warm-start technique. Suppose, that for a given value of C, we have computed the optimal $(d^\star, \alpha^\star)$ pair; the idea of a warm-start is to use this solution for initializing another MKL problem with a different value of C. In our case, we iteratively compute the solutions for decreasing values of C (note that $\alpha^\star$ has to be modified to be a feasible initialization of the more constrained SVM problem).

5. Numerical Experiments

In this experimental section, we essentially aim at illustrating three points. The first point is to show that our gradient descent algorithm is efficient. This is achieved by binary classification experiments, where SimpleMKL is compared to the SILP approach of Sonnenburg et al. (2006). Then, we illustrate the usefulness of a multiple kernel learning approach in the context of regression. The examples we use are based on wavelet-based regression in which the multiple kernel learning framework naturally fits. The final experiment aims at evaluating the multiple kernel approach in a model selection problem for some multiclass problems.

5.1 Computation Time

The aim of this first set of experiments is to assess the running times of SimpleMKL.[2] First, we compare with SILP regarding the time required for computing a single solution of MKL with a given C hyperparameter. Then, we compute an approximate regularization path by varying C values. We finally provide hints on the expected complexity of SimpleMKL, by measuring the growth of running time as the number of examples or kernels increases.

5.1.1 TIME NEEDED FOR REACHING A SINGLE SOLUTION

In this first benchmark, we put SimpleMKL and SILP side by side, for a fixed value of the hyperparameter C ($C = 100$). This procedure, which does not take into account a proper model selection procedure, is not representative of the typical use of SVMs. It is however relevant for the purpose of comparing algorithmic issues.

The evaluation is made on five data sets from the UCI repository: *Liver*, *Wpbc*, *Ionosphere*, *Pima*, *Sonar* (Blake and Merz, 1998). The candidate kernels are:

- Gaussian kernels with 10 different bandwidths σ, on all variables and on each single variable;
- polynomial kernels of degree 1 to 3, again on all and each single variable.

All kernel matrices have been normalized to unit trace, and are precomputed prior to running the algorithms.

Both SimpleMKL and SILP wrap an SVM dual solver based on SimpleSVM, an active constraints method written in Matlab (Canu et al., 2003). The descent procedure of SimpleMKL is

2. All the experiments have been run on a Pentium D-3 GHz with 3 GB of RAM.

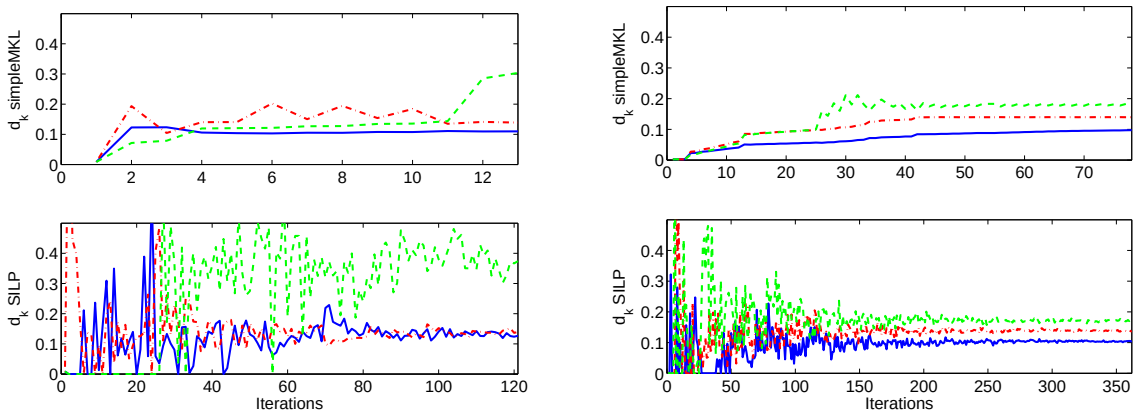


Figure 2: Evolution of the three largest weights d_m for SimpleMKL and SILP; left row: *Pima*; right row: *Ionosphere*.

also implemented in Matlab, whereas the linear programming involved in SILP is implemented by means of the publicly available toolbox LPSOLVE (Berkelaar et al., 2004).

For a fair comparison, we use the same stopping criterion for both algorithms. They halt when, either the duality gap is lower than 0.01, or the number of iterations exceeds 2000. Quantitatively, the displayed results differ from the preliminary version of this work, where the stopping criterion was based on the stabilization of the weights, but they are qualitatively similar (Rakotomamonjy et al., 2007).

For each data set, the algorithms were run 20 times with different train and test sets (70% of the examples for training and 30% for testing). Training examples were normalized to zero mean and unit variance.

In Table 1, we report different performance measures: accuracy, number of selected kernels and running time. As the latter is mainly spent in querying the SVM solver and in computing the gradient of J with respect to d, the number of calls to these two routines is also reported.

Both algorithms are nearly identical in performance accuracy. Their number of selected kernels are of same magnitude, although SimpleMKL tends to select 10 to 20% more kernels. As both algorithms address the same convex optimization problem, with convergent methods starting from the same initialization, the observed differences are only due to the inaccuracy of the solution when the stopping criterion is met. Hence, the trajectories chosen by each algorithm for reaching the solution, detailed in Section 3.4, explain the differences in the number of selected kernels. The updates of d_m based on the descent algorithm of SimpleMKL are rather conservative (small steps departing from $1/M$ for all d_m), whereas the oscillations of cutting planes are likely to favor extreme solutions, hitting the edges of the simplex.

This explanation is corroborated by Figure 2, which compares the behavior of the d_m coefficients through time. The instability of SILP is clearly visible, with very high oscillations in the first iterations and a noticeable residual noise in the long run. In comparison, the trajectories for SimpleMKL are much smoother.

If we now look at the overall difference in computation time reported in Table 1, clearly, on all data sets, SimpleMKL is faster than SILP, with an average gain factor of about 5. Furthermore, the

Liver $\ell = 241$ $M = 91$

Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	10.6 ± 1.3	65.9 ± 2.6	47.6 ± 9.8	99.8 ± 20	99.8 ± 20
SimpleMKL	11.2 ± 1.2	65.9 ± 2.3	18.9 ± 12.6	522 ± 382	37.0 ± 26
Grad. Desc.	11.6 ± 1.3	66.1 ± 2.7	31.3 ± 14.2	972 ± 630	103 ± 27

Pima $\ell = 538$ $M = 117$

Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	11.6 ± 1.0	76.5 ± 2.3	224 ± 37	95.6 ± 13	95.6 ± 13
SimpleMKL	14.7 ± 1.4	76.5 ± 2.6	79.0 ± 13	314 ± 44	24.3 ± 4.8
Grad. Desc.	14.8 ± 1.4	75.5 ± 2.5	219 ± 24	873 ± 147	118 ± 8.7

Ionosphere $\ell = 246$ $M = 442$

Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	21.6 ± 2.2	91.7 ± 2.5	535 ± 105	403 ± 53	403 ± 53
SimpleMKL	23.6 ± 2.6	91.5 ± 2.5	123 ± 46	1170 ± 369	64 ± 25
Grad. Desc.	22.9 ± 3.2	92.1 ± 2.5	421 ± 61.9	4000 ± 874	478 ± 38

Wpbc $\ell = 136$ $M = 442$

Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	13.7 ± 2.5	76.8 ± 1.2	88.6 ± 32	157 ± 44	157 ± 44
SimpleMKL	15.8 ± 2.4	76.7 ± 1.2	20.6 ± 6.2	618 ± 148	24 ± 10
Grad. Desc.	16.8 ± 2.8	76.9 ± 1.5	106 ± 6.1	2620 ± 232	361 ± 16

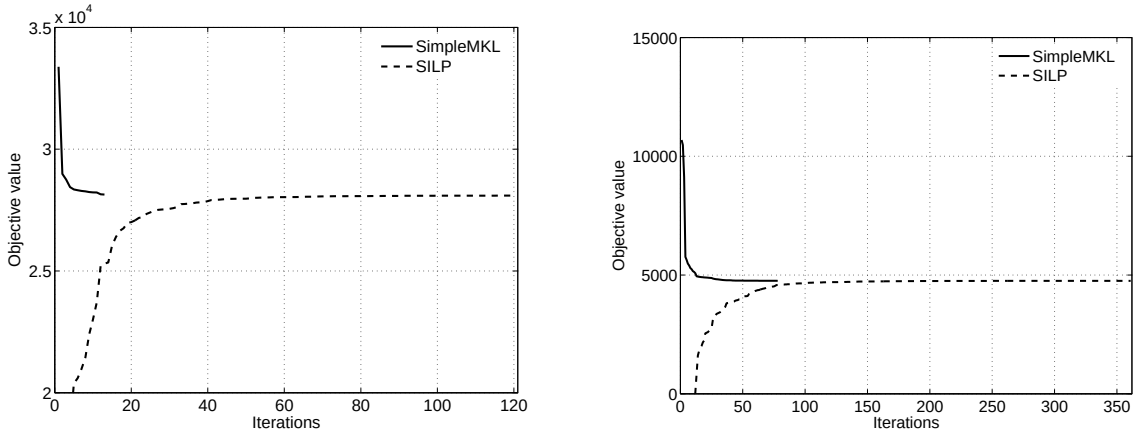
Sonar $\ell = 146$ $M = 793$

Algorithm	# Kernel	Accuracy	Time (s)	# SVM eval	# Gradient eval
SILP	33.5 ± 3.8	80.5 ± 5.1	2290 ± 864	903 ± 187	903 ± 187
SimpleMKL	36.7 ± 5.1	80.6 ± 5.1	163 ± 93	2770 ± 1560	115 ± 66
Grad. Desc.	35.7 ± 3.9	80.2 ± 4.7	469 ± 90	7630 ± 2600	836 ± 99

Table 1: Average performance measures for the two MKL algorithms and a plain gradient descent algorithm.

larger the number of kernels is, the larger the speed gain we achieve. Looking at the last column of Table 1, we see that the main reason for improvement is that SimpleMKL converges in fewer iterations (that is, gradient computations). It may seem surprising that this gain is not counterbalanced by the fact that SimpleMKL requires many more calls to the SVM solver (on average, about 4 times). As we stated in Section 3.4, when the number of kernels is large, computing the gradient may be expensive compared to SVM retraining with warm-start techniques.

To understand why, with this large number of calls to the SVM solver, SimpleMKL is still much faster than SILP, we have to look back at Figure 2. On the one hand, the large variations in subsequents d_m values for SILP, entail that subsequent SVM problems are not likely to have similar solutions: a warm-start call to the SVM solver does not help much. On the other hand, with the smooth trajectories of d_m in SimpleMKL, the previous SVM solution is often a good guess for the



Pima *Ionosphere*

Figure 3: Evolution of the objective values for SimpleMKL and SILP; left row: *Pima*; right row: *Ionosphere*.

current problem: a warm-start call to the SVM solver results in much less computation than a call from scratch.

Table 1 also shows the results obtained when replacing the update scheme described in Algorithm 1 by a usual reduced gradient update, which, at each iteration, modifies d by computing the optimal step size on the descent direction D (12). The training of this variant is considerably slower than SimpleMKL and is only slightly better than SILP. We see that the gradient descent updates require many more calls to the SVM solver and a number of gradient computations comparable with SILP. Note that, compared to SILP, the numerous additional calls to the SVM solver have not a drastic effect on running time. The gradient updates are stable, so that they can benefit from warm-start contrary to SILP.

To end this first series of experiments, Figure 3 depicts the evolution of the objective function for the data sets that were used in Figure 2. Besides the fact that SILP needs more iterations for achieving a good approximation of the final solution, it is worth noting that the objective values rapidly reach their steady state while still being far from convergence, when d_m values are far from being settled. Thus, monitoring objective values is not suitable to assess convergence.

5.1.2 TIME NEEDED FOR GETTING AN APPROXIMATE REGULARIZATION PATH

In practice, the optimal value of C is unknown, and one has to solve several SVM problems, spanning a wide range of C values, before choosing a solution according to some model selection criterion like the cross-validation error. Here, we further pursue the comparison of the running times of SimpleMKL and SILP, in a series of experiments that include the search for a sensible value of C.

In this new benchmark, we use the same data sets as in the previous experiments, with the same kernel settings. The task is only changed in the respect that we now evaluate the running times needed by both algorithms to compute an approximate regularization path.

For both algorithms, we use a simple warm-start technique, which consists in using the optimal solutions $\{d_m^\star\}$ and $\{\alpha_i^\star\}$ obtained for a given C to initialize a new MKL problem with $C + \Delta C$

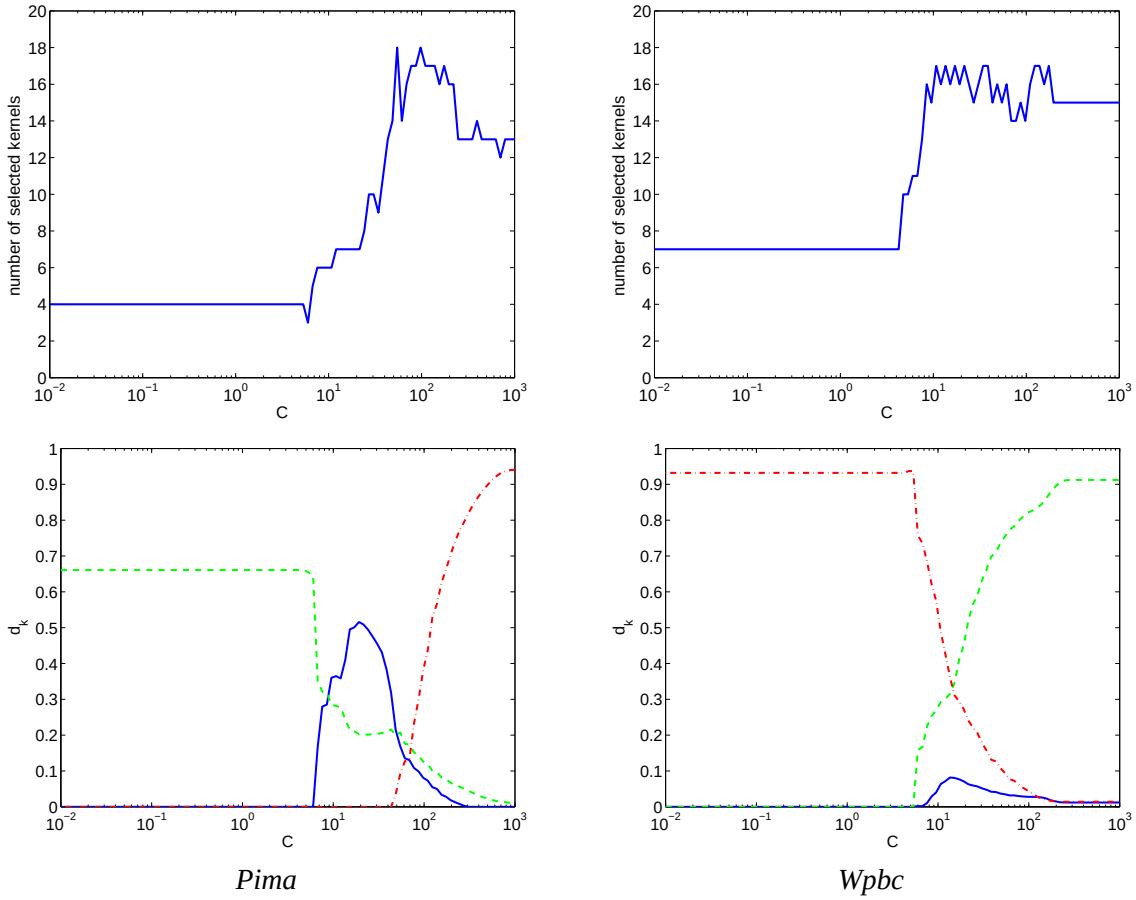


Figure 4: Regularization paths of some d_m and the number of selected kernels versus C; left row: *Pima*; right row: *Wpbc*.

(DeCoste and Wagstaff., 2000). As described in Section 4.4, we start from the largest C and then approximate the regularization path by decreasing its value. The set of C values is obtained by evenly sampling the interval $[0.01, 1000]$ on a logarithmic scale.

Figure 4 shows the variations of the number of selected kernels and the values of d along the regularization path for the *Pima* and *Wpbc* data sets. The number of kernels is not a monotone function of C: for small values of C, the number of kernels is somewhat constant, then, it rises rapidly. There is a small overshoot before reaching a plateau corresponding to very high values of C. This trend is similar for the number of leading terms in the kernel weight vector d. Both phenomenon were observed consistently over the data sets we used.

Table 2 displays the average computation time (over 10 runs) required for building the approximate regularization path. As previously, SimpleMKL is more efficient than SILP, with a gain factor increasing with the number of kernels in the combination. The range of gain factors, from 5.9 to 23, is even more impressive than in the previous benchmark. SimpleMKL benefits from the continuity of solutions along the regularization path, whereas SILP does not take advantage of warm starts. Even provided with a good initialization, it needs many cutting planes to stabilize.

Data Set	SimpleMKL	SILP	Ratio
Liver	148 ± 37	875 ± 125	5.9
Pima	1030 ± 195	6070 ± 1430	5.9
Ionosphere	1290 ± 927	8840 ± 1850	6.8
Wpbc	88 ± 16	2040 ± 544	23
Sonar	625 ± 174	$1.52 \cdot 10^5$ (*)	243

Table 2: Average computation time (in seconds) for getting an approximate regularization path. For the *Sonar* data set, SILP was extremely slow, so that regularization path was computed only once.

5.1.3 MORE ON SIMPLEMKL RUNNING TIMES

Here, we provide an empirical assessment of the expected complexity of SimpleMKL on different data sets from the UCI repository. We first look at the situation where kernel matrices can be pre-computed and stored in memory, before reporting experiments where the memory are too high, leading to repeated kernel evaluations.

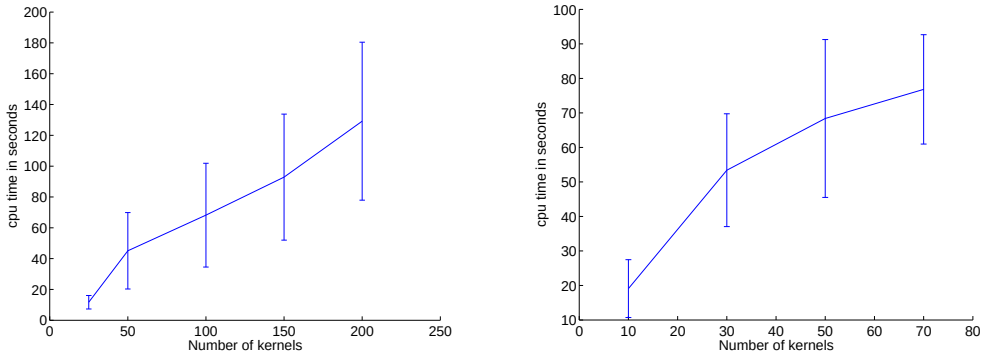
In a first set of experiments, we use Gaussian kernels, computed on random subsets of variables and with random width. These kernels are precomputed and stored in memory, and we report the average CPU running times obtained from 20 runs differing in the random draw of training examples. The stopping criterion is the same as in the previous section: a relative duality gap less than $\varepsilon = 0.01$.

The first two rows of Figure 5 depicts the growth of computation time as the number of kernel increases. We observe a nearly linear trend for the four learning problems. This growth rate could be expected considering the linear convergence property of gradient techniques, but the absence of overhead is valuable.

The last row of Figure 5 depicts the growth of computation time as the number of examples increases. Here, the number of kernels is set to 10. In these plots, the observed trend is clearly superlinear. Again, this trend could be expected, considering that SVM expected training times are superlinear in the number of training examples. As we already mentioned, the complexity of SimpleMKL is tightly linked to the one of SVM training (for some examples of single kernel SVM running time, one can refer to the work of Loosli and Canu 2007).

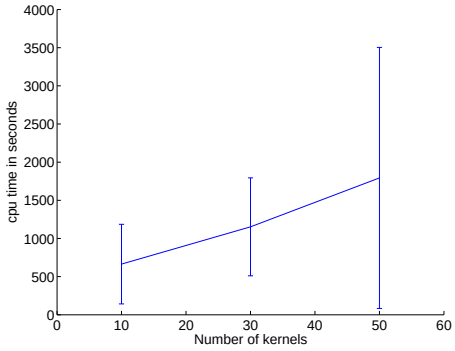
When all the kernels used for MKL cannot be stored in memory, one can resort to a decomposition method. Table 3 reports the average computation times, over 10 runs, in this more difficult situation. The large-scale SVM scheme of Joachims (1999) has been implemented, with basis kernels recomputed whenever needed. This approach is computationally expensive but goes with no memory limit. For these experiments, the stopping criterion is based on the variation of the weights d_m. As shown in Figure 2, the kernel weights rapidly reach a steady state and many iterations are spent for fine tuning the weight and reach the duality gap tolerance. Here, we trade the optimality guarantees provided by the duality gap for substantial computational time savings. The algorithm terminates when the kernel weights variation is lower than 0.001.

Results reported in Table 3 just aim at showing that medium and large-scale situations can be handled by SimpleMKL. Note that Sonnenburg et al. (2006) have run a modified version of their SILP algorithm on a larger scale data sets. However, for such experiments, they have taken advantage of some specific feature map properties. And, as they stated, for general cases where

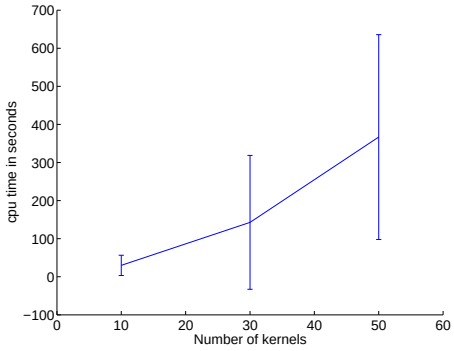


Credit, $\ell = 588$

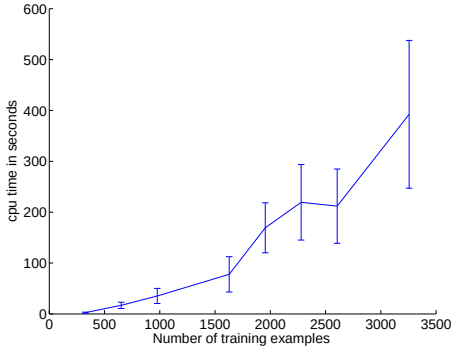
Yeast, $\ell = 1187$



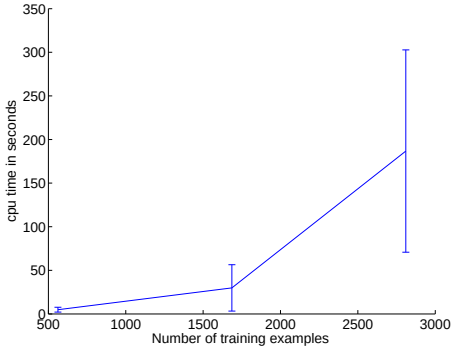
Spamdata, $\ell = 1380$



Optdigits, $\ell = 1686$



Spamdata, $M = 10$



Optdigits, $M = 10$

Figure 5: SimpleMKL average computation times for different data sets; top two rows: number of training examples fixed, number of kernels varying; bottom row: number of training examples varying, number of kernels fixed.

kernel matrices are dense, they have to rely on the SILP algorithm we used in this section for efficiency comparison .

5.2 Multiple Kernel Regression Examples

Several research papers have already claimed that using multiple kernel learning can lead to better generalization performances in some classification problems (Lanckriet et al., 2004a; Zien

Data Set	Nb Examples	# Kernel	Accuracy (%)	Time (s)
Yeast	1335	22	77.25	1130
Spamdata	4140	71	93.49	34200

Table 3: Average computation time needed by SimpleSVM using decomposition methods.

and Ong, 2007; Harchaoui and Bach, 2007). This next experiment aims at illustrating this point but in the context of regression. The problem we deal with is a classical univariate regression problem where the design points are irregular (D'Amato et al., 2006). Furthermore, according to Equation (14), we look for the regression function $f(x)$ as a linear combination of functions each belonging to a wavelet based reproducing kernel Hilbert space.

The algorithm we use is a classical SVM regression algorithm with multiple kernels where each kernel is built from a set of wavelets. These kernels have been obtained according to the expression:

$$K(x,x') = \sum_j \sum_s \frac{1}{2^j} \psi_{j,s}(x) \psi_{j,s}(x')$$

where $\psi(\cdot)$ is a mother wavelet and j,s are respectively the dilation and translation parameters of the wavelet $\psi_{j,s}(\cdot)$. The theoretical details on how such kernels can been built are available in D'Amato et al. (2006); Rakotomamonjy and Canu (2005); Rakotomamonjy et al. (2005).

Our hope when using multiple kernel learning in this context is to capture the multiscale structure of the target function. Hence, each kernel involved in the combination should be weighted accordingly to its correlation to the target function. Furthermore, such a kernel has to be built according to the multiscale structure we wish to capture. In this experiment, we have used three different choices of multiple kernels setting. Suppose we have a set of wavelets with $j \in [j_{min}, j_{max}]$ and $s \in\in [s_{min}, s_{max}]$.

First of all, we have build a single kernel from all the wavelets according to the above equation. Then we have created kernels from all wavelets of a given scale (dilation)

$$K_{Dil,J}(x,x') = \sum_{s=s_{min}}^{s_{max}} \frac{1}{2^j} \psi_{J,s}(x) \psi_{J,s}(x') \qquad \forall J \in [j_{min}, j_{max}]$$

and lastly, we have a set of kernels, where each kernel is built from wavelets located at a given scale and given time-location:

$$K_{Dil-Trans,J,\mathcal{S}}(x,x') = \sum_{s=\mathcal{S}} \frac{1}{2^j} \psi_{J,s}(x) \psi_{J,s}(x') \qquad \forall J \in [j_{min}, j_{max}]$$

where $\mathcal{S}$ is a given set of translation parameter. These sets are built by splitting the full translation parameters index in contiguous and non-overlapping index. The mother wavelet we used is a *Symmlet* Daubechies wavelet with 6 vanishing moments. The resolution levels of the wavelet goes from $j_{min} = -3$ to $j_{min} = 6$. According to these settings, we have 10 dilation kernels and 48 dilation-translation kernels.

We applied this MKL SVM regression algorithm to simulated data sets which are well-known functions in the wavelet literature (Antoniadis and Fan, 2001). Each signal length is 512 and a Gaussian independent random has been added to each signal so that the signal to noise ratio is equal to 5. Examples of the true signals and their noisy versions are displayed in Figure 6. Note that

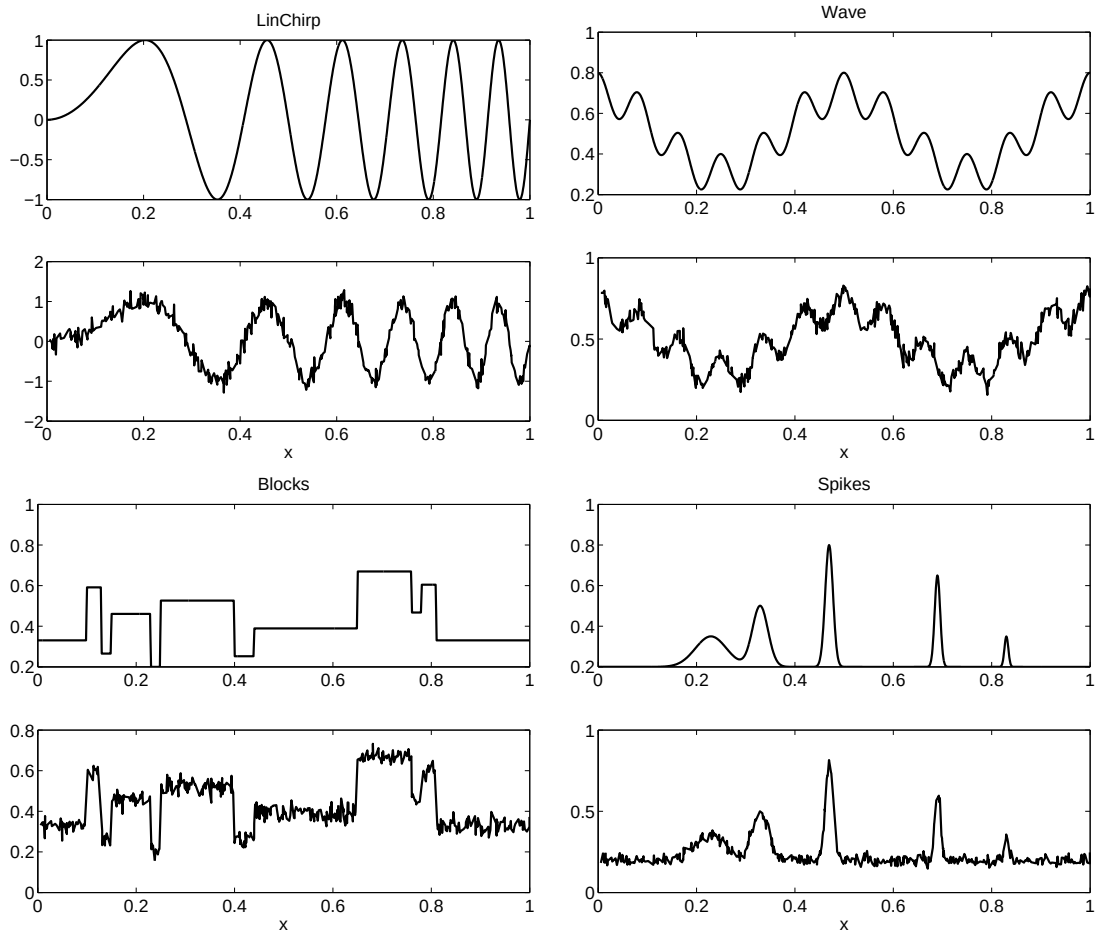


Figure 6: Examples of signals to approximate in the regression problem. (top-left) LinChirp. (top-right) Wave. (bottom-left) Blocks. (bottom-right) Spikes. For each figure, the top plot depicts the true signal while the bottom one presents an example of their randomly sampled noisy versions.

the *LinChirp* and *Wave* signals present some multiscale features that should suit well to an MKL approach.

Performance of the different multiple kernel settings have been compared according to the following experimental setting. For each training signal, we have estimated the regularization parameter C of the MKL SVM regression by means of a validation procedure. The 512 samples have been randomly separated in a learning and a validation sets. Then, by means of an approximate regularization path as described in Section 4.4, we learn different regression functions for 20 samples of C logarithmically sampled on the interval $[0.01, 1000]$. This is performed for 5 random draws of the learning and validation sets. The C value that gives the lowest average normalized mean-square error is considered as the optimal one. Finally, we use all the samples of the training signal and the optimal C value to train an MKL SVM regression. The quality of the resulting regression function is then evaluated with respect to 1000 samples of the true signal. For all the simulations the ε has been fixed to 0.1.

	Single Kernel	Kernel *Dil*		Kernel *Dil-Trans*	
Data Set	Norm. MSE (%)	#Kernel	Norm. MSE	#Kernel	Norm. MSE
LinChirp	1.46 ± 0.28	7.0	1.00 ± 0.15	21.5	0.92 ± 0.20
Wave	0.98 ± 0.06	5.5	0.73 ± 0.10	20.6	0.79 ± 0.07
Blocks	1.96 ± 0.14	6.0	2.11 ± 0.12	19.4	1.94 ± 0.13
Spike	6.85 ± 0.68	6.1	6.97 ± 0.84	12.8	5.58 ± 0.84

Table 4: Normalized Mean Square error for the data described in Figure 6. The results are averaged over 20 runs. The first column give the performance of a SVM regression using a single kernel which is the average sum of all the kernels used for the two other results. Results corresponding to the columns *Kernel Dil* and *Kernel Dil-Trans* are related to MKL SVM regression with multiple kernels. # Kernel denotes the number of kernels selected by SimpleMKL.

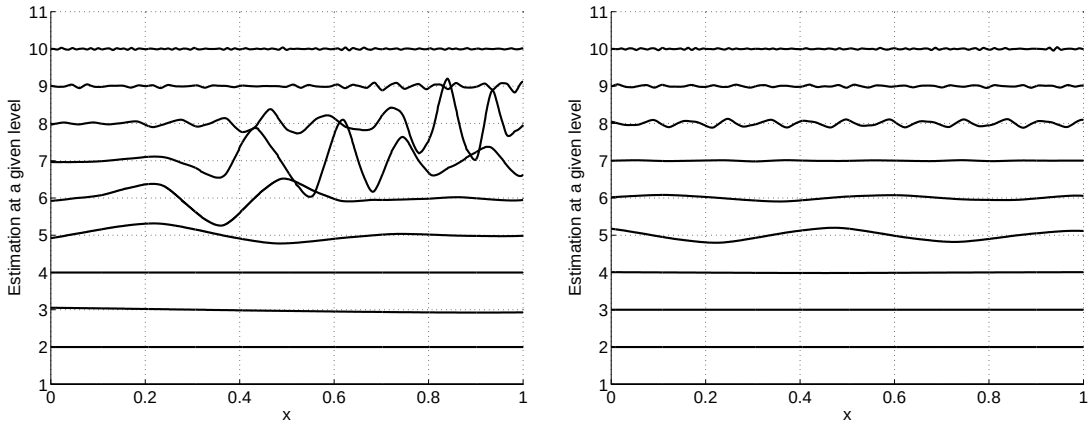


Figure 7: Examples of multiscale analysis of the *LinChirp* signal (left) and the *Wave* signal (right) when using Dilation based multiple kernels. The plots show how each function $f_m(\cdot)$ of the estimation focuses on a particular scale of the target function. The y-axis denotes the scale j of the wavelet used for building the kernel. We can see that some low resolution space are not useful for the target estimation.

Table 4 summarizes the generalization performances achieved by the three different kernel settings. As expected, using a multiple kernel learning setting outperforms the single kernel setting especially when the target function presents multiscale structure. This is noticeable for the *LinChirp* and *Wave* data set. Interestingly, for these two signals, performances of the multiple kernel settings also depend on the signal structure. Indeed, *Wave* presents a frequency located structure while *LinChirp* has a time and frequency located structure. Therefore, it is natural that the Dilation set of kernels performs better than the Dilation-Translation ones for *Wave*. Figure 7 depicts an example of multiscale regression function obtained when using the Dilation set of kernels. These plots show how the kernel weights adapt themselves to the function to estimate. For the same reason of adaptivity to the signal, the Dilation-Translation set of kernels achieves better performances for *Wave* and *Spikes*. We also notice that for the *Blocks* signal using multiple kernels only slightly improves performance compared to a single kernel.

			Training Set Size	
Data Set	#Classes	# examples	Medium	Large
ABE	3	2323	560	1120
DNA	3	3186	500	1000
SEG	7	2310	500	1000
WAV	3	5000	300	600

Table 5: Summary of the multiclass data sets and the training set size used.

	Training set size			
	Medium		Large	
Data Set	MKL	CV	MKL	CV
ABE	0.73 ± 0.28 (16)	0.96 ± 0.36	0.44 ± 0.67 (11)	0.46 ± 0.20
DNA	7.69 ± 0.76 (11)	7.84 ± 0.79	5.59 ± 0.55 (10)	5.59 ± 0.39
SEG	6.52 ± 0.76 (10)	6.51 ± 0.99	4.71 ± 0.67 (13)	4.89 ± 0.71
WAV	15.18 ± 0.90 (15)	15.43 ± 0.97	14.26 ± 0.68 (8)	14.09 ± 0.55

Table 6: Comparison of the generalization performances of an MKL approach and a cross-validation approach for selecting models in some multiclass problems. We have reported the average (over 20 runs) the test set errors of our algorithm while the errors obtained for the SV approach have been extracted from Duan and Keerthi (2005). Results also depend on the training set sizes.

5.3 Multiclass Problem

For selecting the kernel and regularization parameter of a SVM, one usually tries all pairs of parameters and picks the couple that achieves the best cross-validation performance. Using an MKL approach, one can instead let the algorithm combine all available kernels (obtained by sampling the parameter) and just selects the regularization parameter by cross-validation. This last experiment aims at comparing on several multi-class data sets problem, these two model selection approaches (using MKL and CV) for choosing the kernel. Thus, we evaluate the two methods on some multi-class data sets taken from the UCI collection: *dna*, *waveform*, *image segmentation* and *abe* a subset problem of the Letter data set corresponding to the classes A, B and E. Some information about the data set are given in Table 5. For each data set, we divide the whole data into a training set and a test set. This random splitting has been performed 20 times. For ease of comparison with previous works, we have used the splitting proposed by Duan and Keerthi (2005) and available at `http://www.keerthis.com/multiclass.html`. Then we have just computed the performance of SimpleMKL and report their results for the CV approach.

In our MKL *one-against-all* approach, we have used a polynomial kernel of degree 1 to 3 and Gaussian kernel for which σ belongs to $[0.5, 1, 2, 5, 7, 10, 12, 15, 17, 20]$. For the regularization parameter C, we have 10 samples over the interval $[0.01, 10000]$. Note that Duan and Keerthi (2005) have used a more sophisticated sampling strategy based on a coarse sampling of σ and C and followed by fine-tuned sampling procedure. They also select the same couple of C and σ over all pairwise decision functions. Similarly to Duan and Keerthi (2005), the best hyperparameter C has been tuned according to a five-fold cross-validation. According to this best C, we have learned an MKL all the full training set and evaluated the resulting decision function on the test set.

The comparison results are summarized on Table 6. We can see that the generalization performances of an MKL approach is either similar or better than the performance obtained when selecting the kernel through cross-validation, even though we have roughly searched the kernel and regularization parameter space. Hence, we can deduce that MKL can favorably replace cross-validation on kernel parameters. This result based on empirical observations is in accordance with some other works (Lanckriet et al., 2004b; Fung et al., 2004; Kim et al., 2006). However, we think that MKL and thus SimpleMKL in particular, can be better exploited and thus performs better than cross-validation when the kernels have been obtained from heterogeneous source as described for instance in Lanckriet et al. (2004a); Zien and Ong (2007); Harchaoui and Bach (2007).

6. Conclusion

In this paper, we introduced SimpleMKL, a novel algorithm for solving the Multiple Kernel Learning problem. Our formalization of the MKL problem results in a smooth and convex optimization problem, which is actually equivalent to other MKL formulations available in the literature. The main added value of the smoothness of our new objective function is that descent methods become practical and efficient means to solve the optimization problem that wraps a single kernel SVM solver. We provide optimality conditions, analyze convergence and computational complexity issues for binary classification. The SimpleMKL algorithm and the resulting analyses can be easily be transposed to SVM regression, one-class SVM and multiclass SVM to name a few.

We provide experimental evidence that SimpleMKL is significantly more efficient than the state-of-the art SILP approach (Sonnenburg et al., 2006). This efficiency permits to demonstrate the usefulness of our algorithm on wavelet kernel based regression. We also illustrate in multiclass problems that MKL is a viable alternative to cross-validation for selecting a model.

Possible extensions of this work include other learning problems, such as semi-supervised learning or kernel eigenvalue problem like kernel Fisher discriminant analysis. We also plan to explore two different ways to speed up the algorithm. As a first direction, we will investigate ways to obtain a better the descent direction, for example with second-order methods. Note however that computing the Hessian needs the derivative of the dual variable with respects to the weights d. This operation requires solving a linear system (Chapelle et al., 2002) and thus may produce some computational overhead. The second direction is motivated by the observation that most of the computational load is to the computation of the kernel combination. Hence, coordinate-wise optimizers may provide promising routes for improvements.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. Alain Rakotomamonjy, Francis Bach and Stéphane Canu were also supported by French grants from the Agence Nationale de la Recherche (KernSig for AR and SC, MGA for FB).

Appendix A.

This appendix addresses the convexity and differentiability issue of our MKL formulation.

A.1 Proof of Convexity of the Weighted Squared Norm MKL Formulation

The convexity of the MKL problem (2) introduced in Section 2.2 will be established if we prove the convexity of

$$J(f,t) = \frac{1}{t}\langle f, f\rangle_{\mathcal{H}} \quad \text{where } f \in \mathcal{H} \text{ and } t \in \mathbb{R}^{*+}.$$

Since $J(f,t)$ is differentiable with respects to its arguments, we only have to make sure that the first order conditions for convexity are verified. As the convexity of the domain of J is trivial, we verify that, for any (f,t) and $(g,s) \in \mathcal{H} \times \mathbb{R}^{*+}$, the following holds:

$$J(g,s) \geq J(f,t) + \langle \nabla_f J, g-f\rangle_{\mathcal{H}} + (s-t)\nabla_t J \ .$$

As $\nabla_f J = \frac{2}{t} f$ and $\nabla_t J = -\frac{1}{t^2}\langle f, f\rangle_{\mathcal{H}}$, this inequality can be written as

$$\begin{aligned} & \frac{1}{s}\langle g,g\rangle_{\mathcal{H}} \geq \frac{2}{t}\langle f,g\rangle_{\mathcal{H}} - \frac{s}{t^2}\langle f,f\rangle_{\mathcal{H}} \ , \\ \Leftrightarrow \quad & \langle t\,g - s\,f, t\,g - s\,f\rangle_{\mathcal{H}} \geq 0 \ , \end{aligned}$$

where we used that s and t are positive. The above inequality holds since the scalar product on the left-hand-side is a norm. Hence problem (2) minimizes the sum of convex functions on a convex set; it is thus convex. Note that when $\mathcal{H}$ is a finite dimension space, the function $J(f,t)$ is known as the perspective of f, whose convexity is proven in textbooks (Boyd and Vandenberghe, 2004).

A.2 Differentiability of Optimal Value Function

The algorithm we propose for solving the MKL problem heavily relies on the differentiability of the optimal value of the primal SVM objective function. For the sake of self-containedness, we reproduce here a theorem due to Bonnans and Shapiro (1998) that allows us to compute the derivatives of $J(d)$ defined in (8).

Theorem 1 *(Bonnans and Shapiro, 1998) Let X be a metric space and U be a normed space. Suppose that for all $x \in X$ the function $f(x,\cdot)$ is differentiable, that $f(x,u)$ and $D_u f(x,u)$ the derivative of $f(x,\cdot)$ are continuous on $X \times U$ and let Φ be a compact subset of X. Let define the optimal value function as $v(u) = \inf_{x\in\Phi} f(x,u)$. The optimal value function is directionally differentiable. Furthermore, if for $u^0 \in U$, $f(\cdot,u^0)$ has a unique minimizer x^0 over Φ then $v(u)$ is differentiable at u^0 and $Dv(u^0) = D_u f(x^0,u^0)$.*

References

A. Antoniadis and J. Fan. Regularization by wavelet approximations. *J. American Statistical Association*, 96:939–967, 2001.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning, to appear*, 2008.

N. Aronszajn. Theory of reproducing kernels. *Trans. Am. Math. Soc.*, (68):337–404, 1950.

F. Bach. Consistency of the group Lasso and multiple kernel learning. *Journal of Machine Learning Research*, 9:1179–1225, 2008.

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st International Conference on Machine Learning*, pages 41–48, 2004a.

F. Bach, R. Thibaux, and M. Jordan. Computing regularization paths for learning multiple kernels. In *Advances in Neural Information Processing Systems*, volume 17, pages 41–48, 2004b.

M. Berkelaar, K. Eikland, and P. Notebaert. *Lpsolve, Version 5.1.0.0*, 2004. URL `http://lpsolve.sourceforge.net/5.5/`.

D. Bertsekas. *Nonlinear Programming*. Athena scientific, 1999.

C. Blake and C. Merz. UCI repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

F. Bonnans. *Optimisation Continue*. Dunod, 2006.

J.F. Bonnans and A. Shapiro. Optimization problems with pertubation : A guided tour. *SIAM Review*, 40(2):202–227, 1998.

J.F. Bonnans, J.C Gilbert, C. Lemaréchal, and C.A Sagastizbal. *Numerical Optimization Theoretical and Practical Aspects*. Springer, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods Matlab toolbox. LITIS EA4108, INSA de Rouen, Rouen, France, 2003. URL `http://asi.insa-rouen.fr/enseignants/~arakotom/toolbox/index.html`.

C-C. Chang and C-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukerjhee. Choosing multiple parameters for SVM. *Machine Learning*, 46(1-3):131–159, 2002.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

A. D'Amato, A. Antoniadis, and M. Pensky. Wavelet kernel penalized estimation for non-equispaced design regression. *Statistics and Computing*, 16:37–56, 2006.

A. D'Aspremont. Smooth optimization with approximate gradient. *SIAM Journal on Optimization, To appear*, 2008.

D. DeCoste and K. Wagstaff. Alpha seeding for support vector machines. In *International Conference on Knowledge Discovery and Data Mining*, 2000.

K. Duan and S. Keerthi. Which is the best multiclass svm method? an empirical study. In *Multiple Classifier Systems*, pages 278–285, 2005. URL `http://www.keerthis.com/multiclass.html`.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *Annals of statistics*, 32(2):407–499, 2004.

G. Fung, M. Dundar, J. Bi, and B. Rao. A fast iterative algorithm for Fisher discriminant using heterogeneous kernels. In *Proceeedins of the 21th International Conference on Machine Learning*, 2004.

Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In L. Niklasson, M. Bodén, and T. Ziemske, editors, *ICANN'98*, volume 1 of *Perspectives in Neural Computing*, pages 201–206. Springer, 1998.

Y. Grandvalet and S. Canu. Adaptive scaling for feature selection in SVMs. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2003.

Y. Grandvalet and S. Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 445–451. MIT Press, 1999.

Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5:1391–1415, 2004.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

T. Joachims. Making large-scale SVM learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advanced in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press, 1999.

S.-J. Kim, A. Magnani, and S. Boyd. Optimal kernel selection in kernel Fisher discriminant analysis. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

G. Lanckriet, T. De Bie, N. Cristianini, M. Jordan, and W. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20:2626–2635, 2004a.

G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004b.

C. Lemaréchal and C. Sagastizabal. Practical aspects of moreau-yosida regularization : theoretical preliminaries. *SIAM Journal of Optimization*, 7:867–895, 1997.

G. Loosli and S. Canu. Comments on the "Core vector machines: Fast SVM training on very large data sets". *Journal of Machine Learning Research*, 8:291–301, February 2007.

G. Loosli, S. Canu, S. Vishwanathan, A. Smola, and M. Chattopadhyay. Boîte à outils SVM simple et rapide. *Revue d'Intelligence Artificielle*, 19(4-5):741–767, 2005.

D. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.

C. Micchelli and M. Pontil. Learning the kernel function via regularization. *Journal of Machine Learning Research*, 6:1099–1125, 2005.

A. Rakotomamonjy and S. Canu. Frames, reproducing kernels, regularization and learning. *Journal of Machine Learning Research*, 6:1485–1515, 2005.

A. Rakotomamonjy, X. Mary, and S. Canu. Non parametric regression with wavelet kernels. *Applied Stochastics Model for Business and Industry*, 21(2):153–163, 2005.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. More efficiency in multiple kernel learning. In Zoubin Ghahramani, editor, *Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007)*, pages 775–782. Omnipress, 2007.

R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

S. Rosset. Tracking curved regularized optimization solution paths. In *Advances in Neural Information Processing Systems*, 2004.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

S. Sonnenburg, G. Rätsch, and C. Schäfer. A general and efficient algorithm for multiple kernel learning. In *Advances in Neural Information Processing Systems*, volume 17, pages 1–8, 2005.

S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7(1):1531–1565, 2006.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

V. Vapnik, S. Golowich, and A. Smola. Support vector method for function estimation, regression estimation and signal processing. volume Vol. 9. MIT Press, Cambridge, MA, neural information processing systems, edition, 1997.

S. V. N. Vishwanathan, A. J. Smola, and M. Murty. SimpleSVM. In *International Conference on Machine Learning*, 2003.

G. Wahba. *Spline Models for Observational Data*. Series in Applied Mathematics, Vol. 59, SIAM, 1990.

J. Weston and C. Watkins. Multiclass support vector machines. In *Proceedings of ESANN99, Brussels*. D. Facto Press, 1999.

A. Zien and C.S. Ong. Multiclass multiple kernel learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 1191–1198, 2007.

Journal of Machine Learning Research 9 (2008) 2523-2547 Submitted 6/07; Revised 2/08; Published 11/08

Active Learning of Causal Networks with Intervention Experiments and Optimal Designs

Yang-Bo He HEYB@MATH.PKU.EDU.CN
Zhi Geng ZGENG@MATH.PKU.EDU.CN
School of Mathematical Sciences, LMAM
Peking University
Beijing 100871, China

Editor: Andre Elisseeff

Abstract

The causal discovery from data is important for various scientific investigations. Because we cannot distinguish the different directed acyclic graphs (DAGs) in a Markov equivalence class learned from observational data, we have to collect further information on causal structures from experiments with external interventions. In this paper, we propose an active learning approach for discovering causal structures in which we first find a Markov equivalence class from observational data, and then we orient undirected edges in every chain component via intervention experiments separately. In the experiments, some variables are manipulated through external interventions. We discuss two kinds of intervention experiments, randomized experiment and quasi-experiment. Furthermore, we give two optimal designs of experiments, a batch-intervention design and a sequential-intervention design, to minimize the number of manipulated variables and the set of candidate structures based on the minimax and the maximum entropy criteria. We show theoretically that structural learning can be done locally in subgraphs of chain components without need of checking illegal v-structures and cycles in the whole network and that a Markov equivalence subclass obtained after each intervention can still be depicted as a chain graph.

Keywords: active learning, causal networks, directed acyclic graphs, intervention, Markov equivalence class, optimal design, structural learning

1. Introduction

A directed acyclic graph (DAG) (also called a Bayesian network) is a powerful tool to describe a large complex system in various scientific investigations, such as bioinformatics, epidemiology, sociology and business (Pearl, 1988; Lauritzen, 1996; Whittaker, 1990; Aliferis et al., 2003; Jansen et al., 2003; Friedman, 2004). A DAG is also used to describe causal relationships among variables. It is crucial to discover the structure of a DAG for understanding a large complex system or for doing uncertainty inference on it (Cooper and Yoo, 1999; Pearl, 2000). There are many methods of structural learning, and the main methods are Bayesian methods (Cooper and Yoo, 1999; Heckerman, 1997) and constraint-based methods (Spirtes et al., 2000). From data obtained in observational studies, we may not have enough information to discover causal structures completely, but we can obtain only a Markov equivalence class. Thus we have to collect further information of causal structures via experiments with external interventions. Heckerman et al. (1995) discussed structural learning of Bayesian networks from a combination of prior knowledge and statistical data. Cooper and Yoo (1999) presented a method of causal discovery from a mixture of experimental and obser-

vational data. Tian and Pearl (2001a,b) proposed a method of discovering causal structures based on dynamic environment. Tong and Koller (2001) and Murphy (2001) discussed active learning of Bayesian network structures with posterior distributions of structures based on decision theory. In these methods, causal structures are discovered by using additional information from domain experts or experimental data.

Chain graphs were introduced as a natural generalization of DAGs to admit more flexible causal interpretation (Lauritzen and Richardson, 2002). A chain graph contains both directed and undirected edges. A chain component of a chain graph is a connected undirected graph obtained by removing all directed edges from the chain graph. Andersson et al. (1997) showed that DAGs in a Markov equivalence class can be represented by a chain graph. He et al. (2005) presented an approach of structural learning in which a Markov equivalence class of DAGs is sequentially refined into some smaller subclasses via domain knowledge and randomized experiments.

In this paper, we discuss randomized experiments and quasi-experiments of external interventions. We propose a method of local orientations in every chain component, and we show theoretically that the method of local orientations does not create any new v-structure or cycle in the whole DAG provided that neither v-structure nor cycle is created in any chain component. Thus structural learning can be done locally in every chain component without need of checking illegal v-structures and cycles in the whole network. Then we propose the optimal designs of interventional experiments based on the minimax and maximum entropy criteria. These results greatly extend the approach proposed by He et al. (2005). In active learning, we first find a Markov equivalence class from observational data, which can be represented by a chain graph, and then we orient undirected edges via intervention experiments. Two kinds of intervention experiments can be used for orientations. One is randomized experiment, in which an individual is randomly assigned to some level combination of the manipulated variables at a given probability. Randomization can disconnect the manipulated variables from their parent variables in the DAG. Although randomized experiments are most powerful for learning causality, they may be inhibitive in practice. The other is quasi-experiment, in which the pre-intervention distributions of some variables are changed via external interventions, but we cannot ensure that the manipulated variables can be disconnected from their parent variables in the DAG, and thus the post-intervention distributions of manipulated variables may still depend on their parent variables. For example, the pre-intervention distribution of whether patients take a vaccine or not may depend on some variables, and the distribution may be changed by encouraging patients with some benefit in the quasi-experiment, but it may still depend on these variables. Furthermore, we discuss the optimal designs by which the number of manipulated variables is minimized or the uncertainty of candidate structures is minimized at each experiment step based on the minimax and the maximum entropy criteria. We propose two kinds of optimal designs: a batch-intervention experiment and a sequential intervention experiment. For the former, we try to find the minimum set of variables to be manipulated in a batch such that undirected edges are all oriented after the interventions. For the latter, we first choose a variable to be manipulated such that the Markov equivalence class can be reduced by manipulating the variable into a subclass as small as possible, and then according to the current subclass, we repeatedly choose a next variable to be manipulated until all undirected edges are oriented.

In Section 2, we introduce notation and definitions and then show some theoretical results on Markov equivalence classes. In Section 3, we present active learning of causal structures via external interventions and discuss randomized experiments and quasi-experiments. In Section 4, we propose two optimal designs of intervention experiments, a batch-intervention design and a sequen-

tial intervention design. In Section 5, we show simulation results to evaluate the performances of intervention designs proposed in this paper. Conclusions are given in Section 6. Proofs of theorems are given in Appendix.

2. Causal DAGs and Markov Equivalence Class

A graph G can be defined to be a pair $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{V_1, \cdots, V_n\}$ denotes the node set and $\mathbb{E}$ denotes the edge set which is a subset of the set $\mathbb{V} \times \mathbb{V}$ of ordered pairs of nodes. If both ordered pairs (V_i, V_j) and (V_j, V_i) are in $\mathbb{E}$, we say that there is an undirected edge between V_i and V_j, denoted as $V_i - V_j$. If $(V_i, V_j) \in \mathbb{E}$ and $(V_j, V_i) \notin \mathbb{E}$, we call it a directed edge, denoted as $V_i \to V_j$. We say that V_i is a neighbor of V_j if there is an undirected or directed edge between V_i and V_j. A graph is directed if all edges of the graph are directed. A graph is undirected if all edges of the graph are undirected.

A sequence $(V_1, V_2, \cdots, V_k)$ is called a *partially directed path* from V_1 to V_k if either $V_i \to V_{i+1}$ or $V_i - V_{i+1}$ is in G for all $i = 1, \ldots, k-1$. A partially directed path is a directed path if there is not any undirected edge in the path. A node V_i is an *ancestor* of V_j and V_j is a *descendant* of V_i if there is a directed path from V_i to V_j. A *directed cycle* is a directed path from a node to itself, and a *partially directed cycle* is a partially directed path from a node to itself.

A graph with both directed and undirected edges is a chain graph if there is not any partially directed cycle. Figure 1 shows a chain graph with five nodes. A chain component is a node set whose nodes are connected in an undirected graph obtained by removing all directed edges from the chain graph. An undirected graph is chordal if every cycle of length larger than or equal to 4 possesses a chord.

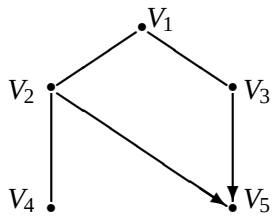


Figure 1: A chain graph G^* depicts the essential graph of G, G_1, G_2 and G_3.

A directed acyclic graph (DAG) is a directed graph which does not contain any directed cycle. A causal DAG is a DAG which is used to describe the causal relationships among variables $V_1, \cdots, V_n$. In the causal DAG, a directed edge $V_i \to V_j$ is interpreted as that the *parent* node V_i is a cause of the *child* node V_j, and that V_j is an effect of V_i. Let $pa(V_i)$ denote the set of all parents of V_i and $ch(V_i)$ denote the set of all *children* of V_i. Let τ be a node subset of $\mathbb{V}$. The *subgraph* $G_\tau = (\tau, \mathbb{E}_\tau)$ induced by the subset τ has the node set τ and the edge set $\mathbb{E}_\tau = \mathbb{E} \cap (\tau \times \tau)$ which contains all edges falling into τ. Two graphs have *the same skeleton* if they have the same set of nodes and the same set of edges regardless of their directions. A head-to-head structure is called a *v-structure* if the parents are not adjacent, such as $V_1 \to V_2 \leftarrow V_3$.

Figure 2 shows four different causal structures of five nodes. The causal graph G in Figure 2 depicts that V_1 is a cause of V_3, which in turn is a cause of V_5.

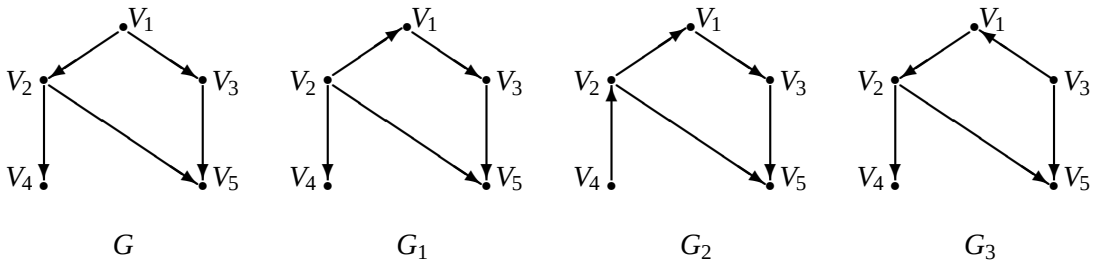


Figure 2: The equivalence class $[G]$.

A joint distribution P satisfies Markov property with respect to a graph G if any variable of G is independent of all its non-descendants in G given its parents with respect to the joint distribution P. Furthermore, the distribution P can be factored as follows

$$P(v_1, v_2, \cdots, v_n) = \prod_{i=1}^{n} P(v_i | pa(v_i)),$$

where v_i denotes a value of variable V_i, and $pa(v_i)$ denotes a value of the parent set $pa(V_i)$ (Pearl, 1988; Lauritzen, 1996; Spirtes et al., 2000). In this paper, we assume that any conditional independence relations in P are entailed by the Markov property, which is called the faithfulness assumption (Spirtes et al., 2000). We also assume that there are no latent variables (that is, no unmeasured variables) in causal DAGs. Different DAGs may encode the same Markov properties. A Markov equivalence class is a set of DAGs that have the same Markov properties. Let $G_1 \sim G_2$ denote that two DAGs G_1 and G_2 are Markov equivalent, and let $[G]$ denote the equivalence class of a DAG G, that is, $[G] = \{G' : G' \sim G\}$. The four DAGs G, G_1, G_2 and G_3 in Figure 2 form a Markov equivalence class $[G]$. Below we review two results about Markov equivalence of DAGs given by Verma and Pearl (1990) and Andersson et al. (1997).

Lemma 1 *(Verma and Pearl, 1990) Two DAGs are Markov equivalent if and only if they have the same skeleton and the same v-structures.*

Andersson et al. (1997) used an essential graph G^* to represent the equivalence class $[G]$.

Definition 2 *The essential graph $G^* = (\mathbb{V}, \mathbb{E}^*)$ of G has the same node set and the same skeleton as G, whose one edge is directed if and only if it has the same orientation in every DAG in $[G]$ and whose other edges are undirected.*

For example, G^* in Figure 1 is the essential graph of G in Figure 2. The edges $V_2 \rightarrow V_5$ and $V_3 \rightarrow V_5$ in G^* are directed since they have the same orientation for all DAGs of $[G]$ in Figure 2, and other edges are undirected.

Lemma 3 *(Andersson et al., 1997) Let G^* be the essential graph of $G = (\mathbb{V}, \mathbb{E})$. Then G^* has the following properties:*

(i) *G^* is a chain graph,*

(ii) *G^*_τ is chordal for every chain component τ, and*

(iii) *$V_i \to V_j - V_k$ does not occur as an induced subgraph of G^*.*

Suppose that G is an unknown underlying causal graph and that its essential graph $G^* = (\mathbb{V}, \mathbb{E})$ has been obtained from observational data, and has k chain components $\{\tau_1, \cdots, \tau_k\}$. Its edge set $\mathbb{E}$ can be partitioned into the set $\mathbb{E}_1$ of directed edges and the set $\mathbb{E}_2$ of undirected edges. Let G^*_τ denote a subgraph of the essential G^* induced by a chain component τ of G^*. Any subgraph of the essential graph induced by a chain component is undirected. Since all v-structures can be discovered from observational data, any subgraph G'_τ of G' should not have any v-structure for $G' \in [G]$. For example, the essential graph G^* in Figure 1 has one chain component $\tau = \{V_1, V_2, V_3, V_4\}$. It can been seen that G'_τ has no v-structure for $G' \in \{G, G_1, G_2, G_3\}$.

Given an essential graph G^*, we need to orient all undirected edges in each chain component to discover the whole causal graph G. Below we show that the orientation can be done separately in every chain component. We also show that there are neither new v-structures nor cycles in the whole graph as long as there are neither v-structures nor cycles in any chain component. Thus in the orientation process, we only need to ensure neither v-structures nor cycles in any component, and we need not check new v-structures and cycles for the whole graph.

Theorem 4 *Let τ be a chain component of an essential graph G^*. For each undirected edge $V - U$ in G^*_τ, neither orientation $V \to U$ nor $V \leftarrow U$ can create a v-structure with any node W outside τ, that is, neither $V \to U \leftarrow W$ nor $W \to V \leftarrow U$ can occur for any $W \notin \tau$.*

Theorem 4 means that there is not any node W outside the component τ which can build a v-structure with two nodes in τ.

Theorem 5 *Let τ be a chain component of G^*. If orientation of undirected edges in the subgraph G^*_τ does not create any directed cycle in the subgraph, then the orientation does not create any directed cycle in the whole DAG.*

According to Theorems 4 and 5, we find that the undirected edges can be oriented separately in each chain component regardless of directed and undirected edges in other part of the essential graph as long as neither cycles nor v-structures are constructed in any chain component. Thus the orientation for one chain component does not affect the orientations for other components. The orientation approach and its correctness will be discussed in Section 3.

3. Active Learning of Causal Structures via External Interventions

To discover causal structures further from a Markov equivalence class obtained from observational data, we have to perform external interventions on some variables. In this section, we consider two kinds of external interventions. One is the randomized experiment, in which the post-intervention distribution of the manipulated variable V_i is independent of its parent variables. The other is the quasi-experiment, in which the distribution of the manipulated variable V_i conditional on its parents $pa(V_i)$ is changed by manipulating V_i. For example, the distribution of whether patients take a vaccine or not is changed by randomly encouraging patients at a discount.

3.1 Interventions by Randomized Experiments

In this subsection, we conduct interventions as randomized experiments, in which some variables are manipulated from external interventions by assigning individuals to some levels of these variables in a probabilistic way. For example, in a clinical trial, every patient is randomly assigned to a treatment group of $V_i = v_i$ at a probability $P'(v_i)$. The randomized manipulation disconnects the node V_i from its parents $pa(V_i)$ in the DAG. Thus the pre-intervention conditional probability $P(v_i|pa(v_i))$ of $V_i = v_i$ given $pa(V_i) = pa(v_i)$ is replaced by the post-intervention probability $P'(v_i)$ while all other conditional probabilities $P(v_j|pa(v_j))$ for $j \neq i$ are kept unchanged in the randomized experiment. Then the post-intervention joint distribution is

$$P_{V_i}(v_1, v_2, \cdots, v_n) = P'(v_i) \prod_{j \neq i} P(v_j|pa(v_j)),$$

(Pearl, 1993). From this post-intervention distribution, we have $P_{V_i}(v_i|pa(v_i)) = P_{V_i}(v_i)$, that is, the manipulated variable V_i is independent of its parents $pa(V_i)$ in the post-intervention distribution. Under the faithfulness assumption, it is obvious that an undirected edge between V_i and its neighbor V_j can be oriented as $V_i \leftarrow V_j$ if the post-intervention distribution has $V_i \perp\!\!\!\perp V_j$, otherwise it is oriented as $V_i \rightarrow V_j$, where $V_i \perp\!\!\!\perp V_j$ denotes that V_i is independent of V_j. The orientation only needs an independence test for the marginal distribution of variables V_i and V_j. Notice that the independence is tested by using only the experimental data without use of the previous observational data.

Let $e(V_i)$ denote the orientation of edges which is determined by manipulating node V_i. If V_i belongs to a chain component τ (that is, it connects at least one undirected edge), then the Markov equivalence class $[G]$ can be reduced by manipulating V_i to the post-intervention Markov equivalence class $[G]_{e(V_i)}$

$$[G]_{e(V_i)} = \{G' \in [G] | G' \text{ has the same orientation as } e(V_i)\}.$$

A Markov equivalence class is split into several subclasses by manipulating V_i, each of which has different orientations $e(V_i)$. Let $G^*_{e(V_i)}$ denote the post-intervention essential graph which depicts the post-intervention Markov equivalence class $[G]_{e(V_i)}$. We show below that $G^*_{e(V_i)}$ also has the properties of essential graphs.

Theorem 6 *Let τ be a chain component of the pre-intervention essential graph G^* and V_i be a node in the component τ. The post-intervention graph $G^*_{e(V_i)}$ is also a chain graph, that is, $G^*_{e(V_i)}$ has the following properties:*

(i) *$G^*_{e(V_i)}$ is a chain graph,*

(ii) *$G^*_{e(V_i)}$ is chordal, and*

(iii) *$V_j \rightarrow V_k - V_l$ does not occur as an induced subgraph of $G^*_{e(V_i)}$.*

By Theorem 6, the pre-intervention chain graph is changed by manipulating a variable to another chain graph which has less undirected edges. Thus variables in chain components can be manipulated repeatedly until the Markov equivalence subclass is reduced to a subclass with a single DAG, and properties of chain graphs are not lost in this intervention process.

According to the above results, we first learn an essential graph from observational data, which is a chain graph (Andersson et al., 1997) and depicts a Markov equivalence class (Heckerman et

al., 1995; Verma and Pearl, 1990; Castelo and Perlman, 2002). Next we choose a variable V_i to be manipulated from a chain component, and we can orient the undirected edges connecting V_i and some other undirected edges whose reverse orientations create v-structures or cycles. Repeating this process, we choose a next variable to be manipulated until all undirected edges are oriented. Below we give an example to illustrate the intervention process.

Example 1. Consider an essential graph in Figure 3, which depicts a Markov equivalence class with 12 DAGs in Figure 4. After obtaining the essential graph from observational data, we manipulate some variables in randomized experiments to identify a causal structure in the 12 DAGs. For example, Table 1 gives four possible orientations and Markov equivalence subclasses obtained by manipulating V_1. A class with 12 DAGs is split into four subclasses by manipulating V_1. The post-intervention subclasses (ii) and (iv) have only a single DAG separately. Notice that undirected edges not connecting V_1 can also be oriented by manipulating V_1. The subclasses (i) and (iii) are depicted by post-intervention essential graphs (a) and (b) in Table 1 respectively, both of which are chain graphs. In Table 2, the first column gives four possible independence sets obtained by manipulating V_1. For the set with $V_1 \perp\!\!\!\perp V_2$ and $V_1 \not\perp\!\!\!\perp V_3$, the causal structure is the DAG (3) in Figure 4, and thus we need not further manipulate other variables. For the third set with $V_1 \not\perp\!\!\!\perp V_2$ and $V_1 \not\perp\!\!\!\perp V_3$, we manipulate the next variable V_2. If $V_2 \perp\!\!\!\perp V_3$, then the causal structure is the DAG (1), otherwise it is the DAG (2). For the fourth set with $V_1 \perp\!\!\!\perp V_2$ and $V_1 \perp\!\!\!\perp V_3$, we may need further to manipulate variables V_2, V_3 and V_4 to identify a causal DAG.

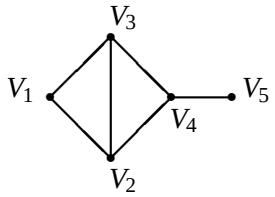


Figure 3: An essential graph of DAGs

3.2 Interventions by Quasi-experiments

In the previous subsection we discussed interventions by randomized experiments. Although randomized experiments are powerful tools to discover causal structures, it may be inhibitive or impractical. In this subsection we consider quasi-experiments. In a quasi-experiment, individuals may choose treatments non-randomly, but their behaviors of treatment choices are influenced by experimenters. For example, some patients may not comply with the treatment assignment from a doctor, but some of them may comply, which is also called an indirect experiment in Pearl (1995).

If we perform an external intervention on V_i such that V_i has a conditional distribution $P'(v_i|pa(v_i))$ different from the pre-intervention distribution $P(v_i|pa(v_i))$ in (1) and other distributions are kept unchanged, then we have the post-intervention joint distribution

$$P_{V_i}(v_1, v_2, \cdots, v_n) = P'(v_i|pa(v_i)) \prod_{j \neq i} P(v_j|pa(v_j)).$$

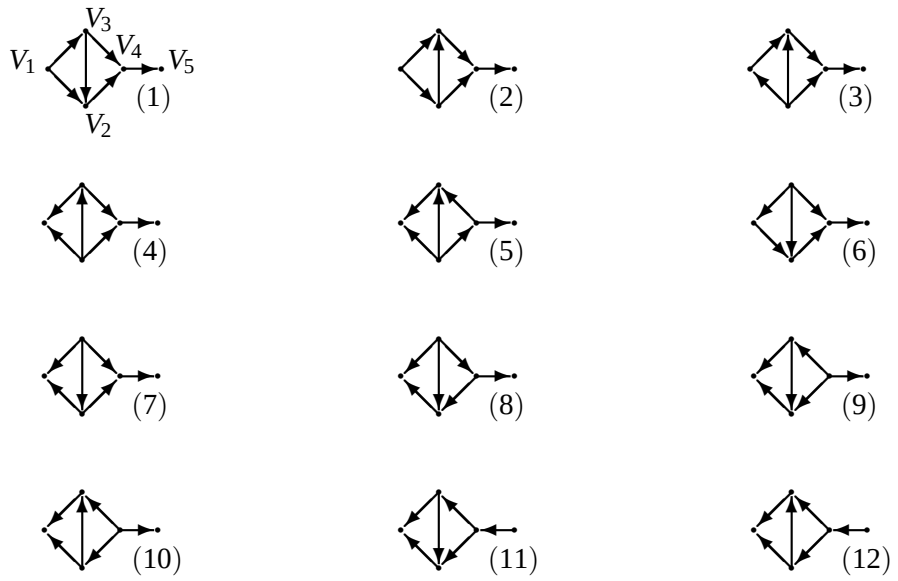


Figure 4: All DAGs in the equivalence class given in Figure 3.

No of subclass	$e(V_1)$	DAGs in a subclass	post-intervention essential graphs
(i)	$V_2 \leftarrow V_1 \rightarrow V_3$	$(1,2)$	V_1 (a)
(ii)	$V_2 \rightarrow V_1 \rightarrow V_3$	(3)	
(iii)	$V_2 \rightarrow V_1 \leftarrow V_3$	$(4,5,$ $7-12)$	V_1 (b)
(iv)	$V_2 \leftarrow V_1 \leftarrow V_3$	(6)	

Table 1: The post-intervention subclasses and essential graphs obtained by manipulating V_1.

In the external intervention, we may not be able to manipulate V_i, but we only need to change its conditional distribution, which may still depend on its parent variables. We call such an experiment a quasi-experiment. Below we discuss how to orient undirected edges via such quasi-experiments. Let τ be a chain component of the essential graph G^*, $ne(V_k)$ be the neighbor set of V_k, C be the children of V_k outside τ (that is, $C = ch(V_k) \setminus \tau$), and B be the set of all potential parents of V_k, that is, $B = ne(V_k) \setminus C$ is the neighbor set of V_k minus the children of V_k which have been identified in the chain graph. Let $V_i - V_k$ be an undirected edge in a chain component τ, and we want to orient the undirected edge by manipulating V_i. Since B is the neighbor set of V_k, we have $V_i \in B$ and thus

V_1	V_2	V_3	V_4	*DAG* in Fig. 4
$V_1 \perp\!\!\!\perp V_2$ *and* $V_1 \not\perp\!\!\!\perp V_3$	*	*	*	(3)
$V_1 \not\perp\!\!\!\perp V_2$ *and* $V_1 \perp\!\!\!\perp V_3$	*	*	*	(6)
$V_1 \not\perp\!\!\!\perp V_2$ *and* $V_1 \not\perp\!\!\!\perp V_3$	$V_2 \perp\!\!\!\perp V_3$	*	*	(1)
	$V_2 \not\perp\!\!\!\perp V_3$	*	*	(2)
$V_1 \perp\!\!\!\perp V_2$ *and* $V_1 \perp\!\!\!\perp V_3$	$V_2 \perp\!\!\!\perp V_3$ *and* $V_2 \not\perp\!\!\!\perp V_4$	*	*	(7)
	$V_2 \not\perp\!\!\!\perp V_3$ *and* $V_2 \not\perp\!\!\!\perp V_4$	$V_3 \not\perp\!\!\!\perp V_4$	*	(4)
		$V_3 \perp\!\!\!\perp V_4$	*	(5)
	$V_2 \perp\!\!\!\perp V_3$ *and* $V_2 \perp\!\!\!\perp V_4$	$V_3 \not\perp\!\!\!\perp V_4$	*	(8)
		$V_3 \perp\!\!\!\perp V_4$	$V_4 \not\perp\!\!\!\perp V_5$	(9)
			$V_4 \perp\!\!\!\perp V_5$	(11)
	$V_2 \not\perp\!\!\!\perp V_3$ *and* $V_2 \perp\!\!\!\perp V_4$	*	$V_4 \not\perp\!\!\!\perp V_5$	(10)
			$V_4 \perp\!\!\!\perp V_5$	(12)

Table 2: The intervention process to identify a causal structure from the essential graph in Figure 3, where $*$ means that the intervention is unnecessary.

$B \neq \emptyset$. Below we show a result which can be used to identify the direction of the undirected edge $V_i - V_k$ via a quasi-experiment of intervention on V_i.

Theorem 7 *For a quasi-experiment of intervention on V_i, we have the following properties*

1. $P_{V_i}(v_k|B) = P(v_k|B)$ *for all v_k and B if V_i is a parent of V_k, and*
2. $P_{V_i}(v_k) = P(v_k)$ *for all v_k if V_i is a child of V_k.*

According to Theorem 7, we can orient the undirected edge $V_i - V_k$ as

1. $V_i \leftarrow V_k$ if $P_{V_i}(v_k|B) \neq P(v_k|B)$ for some v_k and B, or
2. $V_i \rightarrow V_k$ if $P_{V_i}(v_k) \neq P(v_k)$ for some v_k.

The nonequivalence of pre- and post-intervention distributions is tested by using both experimental data and observational data, which is different from that of randomized experiments.

Example 1 (continued). Consider again the essential graph in Figure 3. We use a quasi-experiment of manipulating V_1 in order to orient the undirected edges connecting V_1 ($V_3 - V_1 - V_2$). We may test separately four null hypotheses $P_{V_1}(v_2) = P(v_2)$, $P_{V_1}(v_3) = P(v_3)$, $P_{V_1}(v_2|v_1, v_3, v_4) = P(v_2|v_1, v_3, v_4)$ and $P_{V_1}(v_3|v_1, v_2, v_4) = P(v_3|v_1, v_2, v_4)$ with both observational and experimental data. We orient $V_1 - V_2$ as $V_1 \rightarrow V_2$ if $P_{V_1}(v_2) \neq P(v_2)$, otherwise as $V_1 \leftarrow V_2$ (or further check whether there is a stronger evidence of $P_{V_1}(v_2|v_1, v_3, v_4) \neq P(v_2|v_1, v_3, v_4)$). Similarly we can orient $V_1 - V_3$. Finally we obtain four possible orientations as shown in Table 1.

If both $P_{V_i}(v_k) = P(v_k)$ and $P_{V_i}(v_k|B) = P(v_k|B)$ for all v_k and B hold for a quasi-experiment, then we cannot identify the direction of edge $V_i - V_k$ from the intervention. For example, suppose that there are only two variables V_1 and V_2, V_1 has three levels and V_1 is the parent of V_2. If the true conditional distribution of V_2 given V_1 is: $p(v_2|V_1 = 1) = p(v_2|V_1 = 2) \neq p(v_2|V_1 = 3)$, then the

undirected edge $V_1 - V_2$ cannot be oriented with the intervention on V_1 with $p_{V_1}(V_1 = v) \neq p(V_1 = v)$ for $v = 1$ and 2 but $p_{V_1}(V_1 = 3) = p(V_1 = 3)$ because we have that $p_{V_1}(v_2) = p(v_2)$ for all v_2 and that $p_{V_1}(v_2|B) = p(v_2|B)$ where $B = \{V_1\}$. In a quasi-experiment, an experimenter may not be able to manipulate V_1, and thus this phenomenon can occur. If V_1 can be manipulated, then the experimenter can choose the distribution of V_2 to avoid this phenomenon.

4. Optimal Designs of Intervention Experiments

In this section, we discuss the optimal designs of intervention experiments which are used to minimize the number of manipulated variables or to minimize the uncertainty of candidate structures after an intervention experiment based on some criteria. Since the orientation for one chain component is unrelated to the orientations for other components, we can design an intervention experiment for each chain component separately. As shown in Section 2, given a chain component τ, we orient the subgraph over τ into a DAG G_τ without any v-structure or cycle via experiments of interventions in variables in τ. For simplicity, we omit the subscript τ in this section. In the following subsections, we discuss intervention designs for only one chain component. We first introduce the concept of sufficient interventions and discuss their properties of sufficient interventions, then we present the optimal design of batch interventions, and finally we give the optimal design of sequential interventions. For optimizing quasi-experiments of interventions, we assume that intervention on a variable V_i will change the marginal distribution of its child V_j, that is, there is a level v_j such that $P_{V_i}(v_j) \neq P(v_j)$ for $V_i \to V_j$. Under this assumption, all undirected edges connecting a node V_i can be oriented via a quasi-experiment of intervention on variable V_i. Without the assumption, there may be some undirected edge which cannot be oriented even if we perform interventions in both of its two nodes.

4.1 Sufficient Interventions

It is obvious that we can identify a DAG in a Markov equivalence class if we can manipulate all variables which connect undirected edges. However, it may be unnecessary to manipulate all of these variables. Let $\mathcal{S} = (V_1, V_2, \cdots, V_k)$ denote a sequence of manipulated variables. We say that a sequence of manipulated variables is sufficient for a Markov equivalence class $[G]$ if we can identify one DAG from all possible DAGs in $[G]$ after these variables in the sequence are manipulated. That is, we can orient all undirected edges of the essential graph G^* no matter which G in $[G]$ is the true DAG. There may be several sufficient sequences for a Markov equivalence class $[G]$.

Let g denote the number of nodes in the chain component, and h the number of undirected edges within the component. Then there are at most 2^h possible orientation of these undirected edges, and thus there are at most 2^h DAGs over the component in the Markov equivalence class. Given a permutation of nodes in the component, a DAG can be obtained by orienting all undirected edges backwards in the direction of the permutation, and thus there are at most $\min\{2^h, g!\}$ DAGs in the class.

Theorem 8 *If a sequence $\mathcal{S} = (V_1, V_2, \cdots, V_k)$ of manipulated variables is sufficient, then any permutation of $\mathcal{S}$ is also sufficient.*

According to Theorem 8, we can ignore the order of variables in an intervention sequence and treat the sequence as a variable set. Thus, if $\mathcal{S}$ is a sufficient set, then $\mathcal{S}'$ which contains $\mathcal{S}$

is also sufficient. Manipulating V_i, we obtain a class $E(V_i) = \{e(V_i)\}$ of orientations (see Table 1 as an example). Given an orientation $e(V_i)$, we can obtain the class $[G]_{e(V_i)}$ by (3). We say that $e(V_1,\ldots,V_k) = \{e(V_1),\ldots,e(V_k)\}$ is a legal combination of orientations if there is not any v-structure or cycle formed and there is not any undirected edge oriented in two different directions by these orientations. For a set $\mathcal{S} = (V_1,\ldots,V_k)$ of manipulated variables, the Markov equivalence class is reduced into a class

$$[G]_{e(V_1,\ldots,V_k)} = [G]_{e(V_1)} \cap \ldots \cap [G]_{e(V_k)}$$

for a legal combination $e(V_1,\ldots,V_k)$ of orientations. If $[G]_{e(V_1,\ldots,V_k)}$ has only one DAG for all possible legal combinations $e(V_1,\ldots,V_k) \in E(V_1) \times \ldots \times E(V_k)$, then the set $\mathcal{S}$ is a sufficient set for identifying any DAG in $[G]$. Let $\mathbb{S}$ denote the class of all sufficient sets, that is, $\mathbb{S} = \{\mathcal{S} : \mathcal{S} \text{ is sufficient}\}$. We say that a sequence $\mathcal{S}$ is minimum if any subset of $\mathcal{S}$ is not sufficient.

Theorem 9 *The intersection of all sufficient sets is an empty set, that is, $\bigcap_{\mathcal{S}\in\mathbb{S}} \mathcal{S} = \varnothing$. In addition, the intersection of all minimum sufficient sets is also an empty set.*

From Theorem 9, we can see that there is not any variable that must be manipulated to identify a causal structure. Especially, any undirected edge can be oriented by manipulating either of its two nodes.

4.2 Optimization for Batch Interventions

We say that an intervention experiment is a batch-intervention experiment if all variables in a sufficient set $\mathcal{S}$ are manipulated in a batch to orient all undirected edges of an essential graph. Let $|\mathcal{S}|$ denote the number of variables in $\mathcal{S}$. We say that a batch intervention design is optimal if its sufficient set $\mathcal{S}_o$ has the smallest number of manipulated variables, that is, $|\mathcal{S}_o| = \min\{|\mathcal{S}| : \mathcal{S} \in \mathbb{S}\}$. Given a Markov equivalence class $[G]$, we try to find a sufficient set S which has the smallest number of manipulated variables for identifying all possible DAGs in the class $[G]$. Below we give an algorithm to find the optimal design for batch interventions, in which we first try all sets with a single manipulated variable, then try all sets with two variables, and so on, until each post-intervention Markov equivalence class has a single DAG.

Given a Markov equivalence class $[G]$, we manipulate a node V and obtain an orientation of some edges, denoted by $e(V)$. The class $[G]$ is split into several subclasses, denoted by $[G]_{e(V)}$ for all possible orientations $e(V)$. Let $[G]_{e(V_1,V_2)}$ denote a subclass with an orientation obtained by manipulating V_1 and V_2. The following algorithm 1 performs exhaustive search for the optimal design of batch interventions. Before calling Algorithm 1, we need to enumerate all DAGs in the class $[G]$, and then we can easily find $[G]_{e(V_i)}$ according to (3). There are at most $\min\{g!, 2^h\}$ DAGs in the class $[G]$, and thus the upper bound of the complexity for enumerating all $\{[G]_{e(V_i)}\}$ is $\min\{g!, 2^h\}$. We may be able to have an efficient method to find all $\{[G]_{e(V_i)}\}$ using the structure of the chain component.

Algorithm 1 Algorithm for finding the optimal designs of batch interventions

Input: A chain graph G induced by a chain component $\tau = \{V_1, \ldots, V_g\}$, and $[G]_{e(V_i)}$ for all $e(V_i)$ and i.

Output: All optimal designs of batch interventions.

```
Initialize the size k of the minimum intervention set as k = 0.
repeat
  Set k = k + 1.
  for all possible variable subsets S = {V_{i_1}, ..., V_{i_k}} do
    if |[G]_{e(S)}| = 1 for all possible legal combination e(S) of orientations then
      return the minimum sufficient set S
    end if
  end for
until find some sufficient sets
```

Algorithm 1 exhaustively searches all combinations of manipulated variables to find the minimum sufficient sets, and its complexity is $O(g!)$, although Algorithm 1 may stop whenever it finds some minimum sets. The calculations in Algorithm 1 are only simple set operations

$$[G]_{e(\mathcal{S})} = [G]_{e(V_{i_1})} \cap \ldots \cap [G]_{e(V_{i_k})},$$

where all $[G]_{e(V_i)}$ have been found before calling Algorithm 1. Notice that a single chain component usually has a size g much less than the total number n of variables. Algorithm 1 is feasible for a mild size g. A more efficient algorithm or a greedy method is needed for a large g and h. In this case, there are too many DAGs to enumerate. We can first take a random sample of DAGs from the class $[G]$ with the simulation method proposed in the next subsection, and then we use the sample approximately to find an optimal design.

A possible greedy approach is to select a node to be first manipulated from the chain component which has the largest number of neighbors such that the largest number of undirected edges are oriented by manipulating it, and then delete these oriented edges. Repeat this process until there is not any undirected edge left. But there are cases where the sufficient set obtained from the greedy method is not minimum.

Example 1 (continued). Consider the essential graph in Figure 3, which depicts a Markov equivalence class with 12 DAGs in Figure 4. From Algorithm 1, we can find that $\{1,2,4\}$, $\{1,3,4\}$, $\{2,3,4\}$ and $\{2,3,5\}$ are all the minimum sufficient sets. The greedy method can obtain the same minimum sufficient sets for this example.

4.3 Optimization for Sequential Interventions

The optimal design of batch interventions presented in the previous subsection tries to find a minimum sufficient set $\mathcal{S}$ before any variable is manipulated, and thus it cannot use orientation results obtained by manipulating the previous variables during the intervention process. In this subsection, we propose an experiment of sequential interventions, in which variables are manipulated sequentially. Let $\mathcal{S}^{(t)}$ denote the set of variables that have been manipulated before step t and $\mathcal{S}^{(0)} = \emptyset$. At step t of the sequential experiment, according to the current Markov equivalence class $[G]_{e(\mathcal{S}^{(t-1)})}$ obtained by manipulating the previous variables in $\mathcal{S}^{(t-1)}$, we choose a variable V to be manipulated

based on some criterion. We consider two criteria for choosing a variable. One is the minimax criterion based on which we choose a variable V such that the maximum size of subclasses $[G]_{e(\mathcal{S}^{(t)})}$ for all possible orientations $e(\mathcal{S}^{(t)})$ is minimized. The other is the maximum entropy criterion based on which we choose a variable V such that the following entropy is maximized for any V in the chain component τ

$$H_V = -\sum_{i=1}^{M} \frac{l_i}{L} \log \frac{l_i}{L},$$

where l_i denotes the number of possible DAGs of the chain component with the ith orientation $e(V)_i$ obtained by manipulating V, $L = \sum_i l_i$ and M is the number of all possible orientations $e(V)_1, \ldots, e(V)_M$ obtained by manipulating V. Based on the maximum entropy criterion, the post-intervention subclasses have sizes as small as possible and they have sizes as equal as possible, which means uncertainty for identifying a causal DAG from the Markov equivalence class is minimized by manipulating V. Below we give two examples to illustrate how to choose variables to be manipulated in the optimal design of sequential interventions based on the two criteria.

Example 1 (continued). Consider again the essential graph in Figure 3, which depicts a Markov equivalence class with 12 DAGs in Figure 4. Tables 3 to 6 show the results for manipulating one of variables V_1, V_2 (symmetry to V_3), V_4 and V_5 respectively in order to distinguish the possible DAGs in Figure 4. The first row in these tables gives possible orientations obtained by manipulating the corresponding variable. The second row gives DAGs obtained by the orientation, where numbers are used to index DAGs in Figure 4. The third row gives the number l_i of DAGs of this chain component for the ith orientation. The entropies for manipulating $V_1, \ldots, V_5$ are 0.9831, 1.7046, 1.7046, 1.3480, 0.4506, respectively. Based on the maximum entropy criterion, we choose variable V_2 or V_3 to be manipulated first. The maximum numbers l_i of DAGs for manipulating one of $V_1, \ldots, V_5$ are 8, 3, 3, 6, 10, respectively. Based on the minimax criterion, we also choose variable V_2 or V_3 to be manipulated first.

Although the same variable V_2 or V_3 is chosen to be manipulated first in the above example, in general, the choice may be different based on the two criteria. The minimax criterion tends to be more conservative, and the entropy criterion tends to be more uniform. For example, consider two interventions for an equivalence class with 10 DAGs: one splits the class into 8 subclasses with the numbers $(l_1, \ldots, l_8) = (1,1,1,1,1,1,1,3)$ of DAGs, the other splits it into 5 subclasses with the numbers of DAGs equal to (2,2,2,2,2). Then the minimax criterion chooses the second intervention, while the maximum entropy criterion chooses the first intervention.

To find the number (l_i for $i = 1, \cdots, M$), we need to enumerate all DAGs in the class $[G]$ and then count the number l_i of DAGs with the same orientations as $e(V)_i$. As discussed in Section 4.2, the upper bound of the complexity for calculating all l_i is $O(\min\{g!, 2^h\})$. Generally the size g of a chain component is much less than the number n of the full variable set and the number h of undirected edges in a chain component is not very large. In the following example, we show a special case with a tree structure, where the calculation is easy.

Example 2. In this example, we consider a special case that a chain component has a tree structure. It does not mean that a DAG is a tree, and it is not uncommon in a chain component (see Figure 1). Since there are no v-structures in any chain component, all undirected edges in a subtree can be oriented as long as we find its root. Manipulating a node V in a tree, we can

Orientation	$V_2 \leftarrow V_1 \rightarrow V_3$	$V_2 \rightarrow V_1 \rightarrow V_3$	$V_2 \rightarrow V_1 \leftarrow V_3$	$V_2 \leftarrow V_1 \leftarrow V_3$
DAGs	$\{1,2\}$	$\{3\}$	$\{4,5,7,8,9,10,11,12\}$	$\{6\}$
l_i	2	1	8	1
Entropy is 0.9831 and maximum l_i is 8				

Table 3: Manipulating V_1

Orientation						
DAGs	$\{8,9,11\}$	$\{10,12\}$	$\{3,4,5\}$	$\{2\}$	$\{1,6\}$	$\{7\}$
l_i	3	2	3	1	2	1
Entropy is 1.7046 and maximum l_i is 3						

Table 4: Manipulating V_2

Orientation					
DAGs	$\{1,2,3,4,6,7\}$	$\{5\}$	$\{8\}$	$\{9,10\}$	$\{11,12\}$
l_i	6	1	1	2	2
Entropy is 1.3480 and maximum l_i is 6					

Table 5: Manipulating V_4

Orientation	$V_4 \rightarrow V_5$	$V_4 \leftarrow V_5$
DAGs	$\{1,2,3,4,5,6,7,8,9,10\}$	$\{11,12\}$
l_i	10	2
Entropy is 0.4506 and maximum l_i is 10		

Table 6: Manipulating V_5

determinate all orientations of edges connecting V, and thus all subtrees that are emitted from V can be oriented, but only one subtree with V as a terminal cannot be oriented. Suppose that node V connects M undirected edges, and let l_i denote the number of nodes in the ith subtree connecting V for $i = 1,\ldots,M$. Since each node in the ith subtree may be the root of this subtree, there are l_i possible orientations for the ith subtree. Thus we have the entropy for manipulating V

$$H_V = -\sum_{i=1}^{M} \frac{l_i}{L} log \frac{l_i}{L}.$$

Consider the chain component $\tau = \{V_1,\ldots,V_4\}$ of the chain graph G^* in Figure 1, which has a tree structure. In Table 7, the first column gives variables to be manipulated, the second column gives possible orientations via the intervention, the third column gives the equivalence subclasses (see Figure 2) for each orientation, the fourth column gives the number l_i of possible DAGs for the ith

orientation and the last column gives the entropy for each intervention. From Table 7, we can see that manipulating V_1 or V_2 has the maximum entropy and the minimax size.

Intervention	Orientation	Subclass of DAGs	l_i	H_V
V_1	$V_2 \leftarrow V_1 \rightarrow V_3$	G	1	1.0397
	$V_2 \rightarrow V_1 \rightarrow V_3$	G_1, G_2	2	
	$V_2 \leftarrow V_1 \leftarrow V_3$	G_3	1	
V_2	$V_4 \leftarrow V_2 \leftarrow V_1$	G, G_3	2	1.0397
	$V_4 \leftarrow V_2 \rightarrow V_1$	G_1	1	
	$V_4 \rightarrow V_2 \rightarrow V_1$	G_2	1	
V_3	$V_1 \rightarrow V_3$	G, G_1, G_2	3	0.5623
	$V_1 \leftarrow V_3$	G_3	1	
V_4	$V_4 \leftarrow V_2$	G, G_1, G_3	3	0.5623
	$V_4 \rightarrow V_2$	G_2	1	

Table 7: Manipulating variables in a chain component with a tree structure.

An efficient algorithm or an approximate algorithm is necessary when both g and h are very large. A simulation algorithm can be used to estimate l_i/L. In this simulation method, we randomly take a sample of DAGs without any v-structure from the class $[G]$. To draw such a DAG, we randomly generate a permutation of all nodes in the class, orient all edges backwards in the direction of the permutation, and keep only the DAG without any v-structure. There may be some DAGs in the sample which are the same, and we keep only one of them. Then we count the number l_i' of DAGs in the sample which have the same orientation as $e(V)_i$. We can use l_i'/L' to estimate l_i/L, where $L' = \sum_i l_i'$. When the sample size tends to infinite, all DAGs in the class can be drawn, and then the estimate l_i'/L' tends to l_i/L. Another way to draw a DAG is that we randomly orient each undirected edge of the essential graph, but we need to check whether there is any cycle besides v-structure.

5. Simulation

In this section, we use two experiments to evaluate the active learning approach and the optimal designs via simulations. In the first experiment, we evaluate a whole process of structural learning and orientation in which we first find an essential graph using the PC algorithm and then orient the undirected edges using the approaches proposed in this paper. In the second experiment, we compare various designs for orientations starting with the same underlying essential graph. For both experiments, the DAG (1) in Figure 4 is used as the underlying DAG and all variables are binary. Its essential graph is given in Figure 3 and there are other 11 DAGs which are Markov equivalent to the underlying DAG (1), as shown in Figure 4. This essential graph can also be seen as a chain component of a large essential graph. All conditional probabilities $P(v_j|pa(v_j))$ are generated from the uniform distribution $U(0,1)$. We repeat 1000 simulations with the sample size $n = 1000$.

In each simulation of the first experiment, we first use the PC algorithm to find an essential graph with the significance level $\alpha = 0.15$ with which the most number of true essential graphs were obtained among various significance levels in our simulations. Then we use the intervention approach proposed in Section 3 to orient undirected edges of the essential graph. To compare the performances of orientations for different significance levels and sample sizes used in

intervention experiments, we run simulations for various combinations of significance levels $\alpha_I = 0.01, 0.05, 0.10, 0.15, 0.20, 0.30$ and sample sizes $n_I = 50, 100, 200, 500$ in intervention experiments. To compare the performance of the experiment designs, we further give the numbers of manipulated variables that are necessary to orient all undirected edges of the same essential graphs in various intervention designs. We run the simulations using R 2.6.0 on an Intel(R) Pentium(R) M Processor with 2.0 GHz and 512MB RAM and MS XP. It takes averagely 0.4 second of the processor time for a simulation, and each simulation needs to finish the following works: (1) generate a joint distribution and then generate a random sample of size $n = 1000$, (2) find an essential graph using the PC algorithm, (3) find an optimal design, and (4) repeatedly generate experimental data of size n_I until identifying a DAG.

To make the post-intervention distribution $P'(v_i|pa(v_i))$ different from the pre-intervention $P(v_i|pa(v_i))$, we use the post-intervention distribution of the manipulated variable V_i as follows

$$P'(v_i|pa(v_i)) = P'(v_i) = \begin{cases} 1, & P(v_i) \leq 0.5; \\ 0, & \text{otherwise.} \end{cases}$$

To orient an undirected edge $V_i - V_j$, we implemented both the independence test of the manipulated V_i and its each neighbor variable V_j for randomized experiments and the equivalence test of pre- and post-intervention distributions (i.e., $P_{V_i}(v_j) = P(v_j)$ for all v_j) in our simulations. Both tests have the similar results and the independence test is little more efficient than the equivalence test. To save space, we only show the simulation results of orientations obtained by the equivalence test and the optimal design based on the maximum entropy criterion in Table 8, and other designs have the similar results of orientations.

To evaluate the performance of orientation, we define the percentage of correct orientations as the ratio of the number of correctly oriented edges to the number of edges that are obtained from the PC algorithm and belong to the DAG (1) in Figure 4. The third column λ in Table 8 shows the average percentages of correctly oriented edges of the DAG (1) in 1000 simulations. To separate the false orientations due to the PC algorithm from those due to intervention experiments, we further check the cases that the essential graph in Figure 3 is correctly obtained from the PC algorithm. The fourth column *m* shows the number of correct essential graphs obtained from the PC algorithm in 1000 simulations. In the fifth column, we show the percentage λ' of correct orientations for the correct essential graph. Both λ and λ' increase as n_I increases. Comparing λ and λ', it can be seen that there are more edges oriented correctly when the essential graph is correctly obtained from the PC algorithm. From the sixth to eleven columns, we give the cumulative distributions of the number of edges oriented correctly when the essential graph is correctly obtained. The column labeled '$\geq i$' means that we correctly oriented more than or equal to *i* of 6 edges of the essential graph in Figure 3, and the values in this column denote the percents of DAGs with more than or equal to *i* edges correctly oriented in those simulations. For example, the column '≥ 5' means that more than or equal to 5 edges are oriented correctly (i.e., the DAGs (1), (2) and (6) in Figure 4), and 0.511 in the first line means that 51.1% of $m = 409$ correct essential graphs were oriented with '≥ 5' correct edges. The column '6' means that the underlying DAG (1) is obtained correctly. From this column, it can be seen that more and more DAGs are identified correctly as the size n_I increases. The cumulative distribution for ≥ 0 is equal to one and is omitted. From these columns, it can be seen that more and more edges are correctly oriented as the size n_I increases. From λ and λ', we can see that a larger α_I is preferable for a smaller size n_I, and a smaller α_I is preferable for a larger

n_I. For example, $\alpha_I = 0.20$ is the best for $n_I = 50$, $\alpha_I = 0.10$ for $n_I = 100$, $\alpha_I = 0.05$ for $n_I = 200$, $\alpha_I = 0.01$ for $n_I = 500$.

					The number of edges oriented correctly					
n_I	α_I	λ	m	λ'	6	≥ 5	≥ 4	≥ 3	≥ 2	≥ 1
50	.01	.672	409	.758	0.401	0.511	0.868	0.870	0.927	0.973
	.05	.699	409	.782	0.496	0.616	0.829	0.839	0.934	0.976
	.10	.735	418	.808	0.538	0.646	0.833	0.868	0.969	0.993
	.15	.745	407	.821	0.516	0.690	0.855	0.909	0.966	0.990
	.20	.756	404	.826	0.564	0.723	0.832	0.899	0.963	0.978
	.30	.741	373	.819	0.501	0.729	0.823	0.920	0.965	0.979
100	.01	.761	401	.850	0.586	0.706	0.910	0.925	0.975	0.995
	.05	.774	408	.846	0.588	0.721	0.885	0.919	0.973	0.993
	.10	.806	425	.878	0.668	0.814	0.896	0.925	0.974	0.993
	.15	.794	410	.868	0.624	0.790	0.878	0.932	0.985	1.000
	.20	.788	382	.875	0.626	0.812	0.890	0.948	0.982	0.992
	.30	.798	417	.861	0.583	0.777	0.856	0.959	0.988	1.000
200	.01	.822	421	.901	0.724	0.808	0.945	0.948	0.988	0.995
	.05	.836	402	.911	0.701	0.853	0.950	0.973	0.995	0.995
	.10	.833	408	.900	0.686	0.863	0.917	0.949	0.993	0.995
	.15	.823	382	.901	0.696	0.851	0.911	0.955	0.995	1.000
	.20	.826	395	.886	0.658	0.820	0.889	0.962	0.990	0.997
	.30	.822	402	.887	0.614	0.828	0.905	0.975	0.998	1.000
500	.01	.870	369	.966	0.878	0.943	0.984	0.992	1.000	1.000
	.05	.869	388	.940	0.802	0.920	0.951	0.977	0.995	0.997
	.10	.863	399	.936	0.762	0.905	0.952	0.995	1.000	1.000
	.15	.859	433	.926	0.723	0.898	0.956	0.986	0.995	1.000
	.20	.846	390	.923	0.703	0.890	0.956	0.990	0.997	1.000
	.30	.834	389	.893	0.599	0.820	0.949	0.992	1.000	1.000

Table 8: The simulation results

In the second experiment, we compare the numbers of manipulated variables to orient the same underlying essential graph for different experimental designs. In the following simulations, we set $n_I = 100$ and $\alpha_I = 0.1$, and all orientations start with the true essential graph in Figure 3. As shown in Section 4.2, the optimal batch design and the design by the greedy method always need three variables to be manipulated for orientation of the essential graph. For the optimal sequential designs, the frequencies of the numbers of manipulated variables in 1000 simulations are given in Table 9. In the random design labeled 'Random', we randomly select a variable to be manipulated at each sequential step, only one variable is manipulated for orientations in 268 of 1000 simulations, and four variables are manipulated in 55 of 1000 simulations. In the middle of Table 9, we show the simulation results of the optimal sequential designs based on the minimax criterion and its approximate designs obtained by drawing a sample of DAGs. The minimax design needs only one or two variables to be manipulated in all 1000 simulations. We show three approximate designs which draw h, $h \times 5$ and $h \times 10$ DAGs from a chain component with h undirected edges respectively. For

example, the sample sizes of DAGs from the initial essential graph $[G]$ with $h = 6$ undirected edges are 6, 30 and 60, respectively. As the sample size increases, the distribution of the manipulated variable numbers tends to the distribution for the exact minimax design. The optimal sequential design based on the maximum entropy criterion has a very similar performance as that based on the minimax criterion, as shown in the bottom of Table 9. According to Table 9, all of the sequential intervention designs (Random, Minimax, Entropy and their approximations) are more efficient than the batch design, and the optimal designs based on the minimax and the maximum entropy criteria are more efficient than the random design.

	m^*			
Design	1	2	3	4
Random	268	475	202	55
Minimax	437	563	0	0
Approx. (h)	372	469	159	0
Approx. ($h \times 5$)	413	573	14	0
Approx. ($h \times 10$)	426	574	0	0
Entropy	441	559	0	0
Approx. (h)	375	454	171	0
Approx. ($h \times 5$)	435	547	18	0
Approx. ($h \times 10$)	425	574	1	0

m^* denotes the number of manipulated variables

Table 9: The frequencies of the numbers of interventions

6. Conclusions

In this paper, we proposed a framework for active learning of causal structures via intervention experiments, and further we proposed optimal designs of batch and sequential interventions based on the minimax and the maximum entropy criteria. A Markov equivalence class can be split into subclasses by manipulating a variable, and a causal structure can be identified by manipulating variables repeatedly. We discussed two kinds of external intervention experiments, the randomized experiment and the quasi-experiment. In a randomized experiment, the distribution of a manipulated variable does not depend on its parent variables, while in a quasi-experiment, it may depend on its parents. For a randomized experiment, the orientations of an undirected edge can be determined by testing the independence of the manipulated variable and its neighbor variable only with experimental data. For a quasi-experiment, the orientations can be determined by testing the equivalence of pre- and post-intervention distributions with both experimental and observational data. We discussed two optimal designs of batch and sequential interventions. For the optimal batch design, a smallest set of variables to be manipulated is found before interventions, which is sufficient to orient all undirected edges of an essential graph. But the optimal batch design does not use orientation results obtained by manipulating the previous variables during the intervention process, and thus it may be less efficient than the optimal sequential designs. For the optimal sequential design, we choose a variable to be manipulated sequentially such that the current Markov equivalence class can be reduced to a subclass with potential causal DAGs as little as possible. We discussed two

criteria for optimal sequential designs, the minimax and the maximum entropy criteria. The exact, approximate and greedy methods are presented for finding the optimal designs.

The scalability of the optimal designs proposed in this paper depends only on the sizes of chain components but does not depend on the size of a DAG since the optimal designs are performed separately within every chain component. As discussed in Section 4, the optimal designs need to find the number of possible DAGs in a chain component, which has a upper bound $\min\{2^h, g!\}$. When both the number h of undirected edges and the number g of nodes in a chain component are very large, instead of using the optimal designs, we may use the approximate designs via sampling DAGs. We checked several standard graphs found at the Bayesian Network Repository (http://compbio.cs.huji.ac.il/Repository/). We extracted their chain components and found that most of their chain components have tree structures and their sizes are not large. For example, ALARM with 37 nodes has 4 chain components with only two nodes in each component, HailFinder with 56 nodes has only one component with 18 nodes, Carpo with 60 nodes has 9 components with at most 7 nodes in each component, Diabets with 413 nodes has 25 components with at most 3 nodes, and Mumin 2 to Mumin 4 with over 1000 nodes have at most 21 components with at most 35 nodes. Moreover, all of those largest chain components have tree structures, and thus we can easily carry out optimal designs as discussed in Example 2.

In this paper, we assume that there are no latent variables. Though the algorithm can orient the edges of an essential graph and output a DAG based on a set of either batch or sequential interventions, the application of the method for learning causality in the real word is pretty limited because latent or hidden variables are typically present in real-world data sets.

Acknowledgments

We would like to thank the guest editors and the three referees for their helpful comments and suggestions that greatly improved the previous version of this paper. This research was supported by Doctoral Program of Higher Education of China (20070001039), NSFC (70571003, 10771007, 10431010), NBRP 2003CB715900, 863 Project of China 2007AA01Z43, 973 Project of China 2007CB814905 and MSRA.

Appendix A. Proofs of Theorems

Before proving Theorems 4 and 5, we first give a lemma which will be used in their proofs.

Lemma 10 *If a node $V \in \mathbb{V}$ is a parent of a node U in a chain component τ of G^* (i.e., $(V \to U) \in G^*$, $U \in \tau$, $V \in \mathbb{V}$ and $V \notin \tau$), then V is a parent of all nodes in τ (i.e., $(V \to W) \in G$ for any $W \in \tau$).*

Proof By (iii) of Lemma 3, $V \to U - W$ does not occur in any induced subgraph of G^*. Thus for any neighbor of U in the chain component τ, W and V must be adjacent in G^*. Because $V \notin \tau$, the edge between V and W is directed. There are two alternatives as shown in Figures 5 and 6 for the subgraph induced by $\{V, U, W\}$.

If it is the subgraph in Figure 6 (i.e., the $V \to W \in G'$ for any $G' \in [G]$), then $W \to U$ must be in G' for any $G' \in [G]$ in order to avoid a directed cycle, as shown in Figure 7. So $W \to U$ must be in G^*. It is contrary to the fact that $\{U, W\} \in \tau$ is in a chain component of G^*. So V must also be a parent of W. Because all variables in τ are connected by undirected edges in G^*_τ, V must be a parent

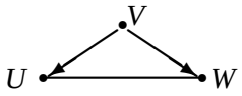


Figure 5: SG_1

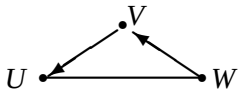


Figure 6: SG_2

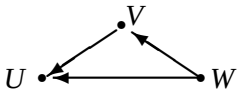


Figure 7: SG_3

of all other variables in τ. ■

Proof of Theorem 4. According to Lemma 10, if a node W outside a component τ points at a node V in τ, then W must point at each node U in τ. Thus W, V and U cannot form a v-structure. ■

Proof of Theorem 5. Suppose that Theorem 5 does not hold, that is, there is a directed path $V_1 \to \cdots \to V_k$ in G_τ which is not a directed cycle, but $W_1 \to \cdots \to W_i \to V_1 \to \cdots \to V_k \to W_{i+1} \to \cdots \to W_1$ is a directed cycle, where $W_i \notin \tau$. We denote this cycle as DC. From Lemma 10, W_i must also be a parent of V_k, and thus $W_1 \to \cdots \to W_i \to V_k \to W_{i+1} \to \cdots \to W_1$ is also a directed cycle, denoted as DC'. Now, every edge of DC' is out of G_τ. Similarly, we can remove all edges in other chain components from DC' and keep the path being a directed cycle. Finally, we can get a directed cycle in the directed subgraph of G^*. It contradicts the fact that G^* is an essential graph of a DAG. So we proved Theorem 5. ■

To prove Theorem 6, we first present an algorithm for finding the post-intervention essential graph $G^*_{e(V)}$ via the orientation $e(V)$, then we show the correctness of the algorithm using several lemmas, and finally we give the proof of Theorem 6 with $G^*_{e(V)}$ obtained by the algorithm. In order to prove that $G^*_{e(V)}$ is also a chain graph, we introduce an algorithm (similar to Step D of SGS and the PC algorithm in Spirtes et al., 2000) for constructing a graph, in which some undirected edges of the initial essential graph are oriented with the information of $e(V)$. Let τ be a chain graph of G^*, $V \in \tau$ and $e(V)$ be an orientation of undirected edges connecting V.

Algorithm 2 Find the post-intervention essential graph via orientation $e(V)$

Input: The essential graph G^* and $e(V)$

Output: The graph H

Orient the undirected edges connecting V in the essential graph G^* according to $e(V)$ and denote the graph as H.

Repeat the following two rules to orient some other undirected edges until no rules can be applied:

(i) if $V_1 \to V_2 - V_3 \in H$ and V_1 and V_3 are not adjacent in H, then orient $V_2 - V_3$ as $V_2 \to V_3$ and update H;

(ii) if $V_1 \to V_2 \to V_3 \in H$ and $V_1 - V_3 \in H$, then orient $V_1 - V_3$ as $V_1 \to V_3$ and update H.

return the graph H

It can be shown that H constructed by Algorithm 2 is a chain graph and H is equal to the post-intervention essential graph $G^*_{e(V)}$. We show those results with the following three Lemmas.

Lemma 11 *Let G^* be the essential graph of DAG G, τ be a chain component of G^* and I be a DAG over τ. Then there is a DAG $G' \in [G]$ such that $I = G'_\tau$ if and only if I is a DAG with the same skeleton as G^*_τ and without v-structures.*

Proof If there is a DAG $G' \in [G]$ such that $I = G'_\tau$, we have from Lemma 1 that I is a DAG with the same skeleton as G^*_τ and without v-structures.

Let I be a DAG with the same skeleton as G^*_τ and without v-structures, and G' be any DAG in the equivalence class $[G]$. We construct a new DAG I' from G' by substituting the subgraph G'_τ of G' with I. I' has the same skeleton as G'. From Theorems 4 and 5, I' has the same v-structures as G'. Thus I' is equivalent to G' and $I' \in [G]$. ■

Lemma 12 *Let H be a graph constructed by Algorithm 2. Then H is a chain graph.*

Proof If H is not a chain graph, there must be a directed cycle in subgraph H_τ for some chain component of G^*. Moreover, G^*_τ is chordal and $H \subset G^*$, and thus H_τ is chordal too. So we can get a three-edge directed cycle in H_τ as given in Figure 8 or 9.

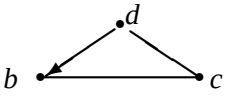


Figure 8: SG_6

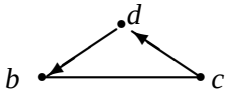


Figure 9: SG_{6_1}

If Figure 9 is a subgraph of H obtained at some step of Algorithm 2, then the undirected edge b—c is oriented as $b \leftarrow c$ according to Algorithm 2. Thus only Figure 8 can be a subgraph of H.

According to Lemma 10, we have that the directed edge $d \rightarrow b$ is not in G^*. Since all edges connecting a have been oriented in Step 1 of Algorithm 2, $d \rightarrow b$ is not an edge connecting a. So $d \rightarrow b$ must be identified at step 2 of Algorithm 2. There are two situations, one is to avoid a v-structure as shown in Figure 10, the other is to avoid a directed cycle as Figure 13.

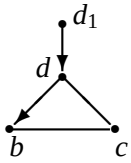


Figure 10: SG_7

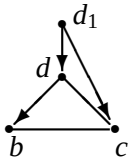


Figure 11: SG_8

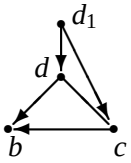


Figure 12: SG_9

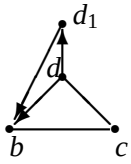


Figure 13: SG_{10}

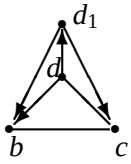


Figure 14: SG_{11}

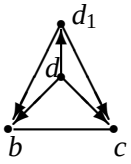


Figure 15: SG_{12}

We can arrange all directed edges in H_τ in order of orientations performed at Step 2 of Algorithm 2. First, we prove that the directed edge $d \rightarrow b$ in Figure 8 is not the first edge oriented at Step 2 of Algorithm 2.

In the first case as Figure 10, if $d \to b$ is the first edge oriented at Step 2 of Algorithm 2, we have $d_1 = a$. Because b and a are not adjacent, and $d—c$ is an undirected edge in H, we have that $d_1 \to c$ must be in H as Figure 11, where $d_1 = a$. Now we consider the subgraph $b—c \leftarrow d_1$. According to the rules (i) and (ii) in Algorithm 2, we have that $b \leftarrow c$ is in $G^*_{e(a)}$ as Figure 12, which contradicts the assumption that $b—c \in H$.

In the second case as Figure 13, if $d \to b$ is the first edge oriented at Step 2 of Algorithm 2, we have $d_1 = a$.

Considering the structure $d_1 \to b—c$ and that $d—c$ is an undirected edge in H, we have that $d_1 \to c$ must be in H as Figure 14. Now we consider the subgraph of $\{d, d_1, c\}$. By Algorithm 2, $d \to c$ is in H as Figure 15, which contradicts the assumption that $d—c \in H$. Thus we have that the first edge oriented at Step 2 of Algorithm 2 is not in any directed cycle. Suppose that the first k oriented edges at Step 2 of Algorithm 2 are not in any directed cycle. Then we want to prove that the $(k+1)th$ oriented edge is also not in a directed cycle.

Let $d \to b$ be the $(k+1)th$ oriented edge at Step 2 of Algorithm 2, and Figure 8 be a subgraph of H. There are also two cases as Figures 10 and 13 for orienting $d \to b$.

In the case of Figure 10, since $d_1 \to d$ is in the first k oriented edges and $d—c \in H$, we have that $d_1 \to c$ must be in H. We also get that $b \leftarrow c$ must be in H as Figure 12, which contradicts the assumption that $b—c \in H$.

In the case of Figure 10, since $d_1 \to b$ and $d \to d_1$ are in the first k oriented edges and $b—c \in H$, we have that $d_1 \to c$ must be in H. We also get that $d \leftarrow c$ must be in H as Figure 15, which contradicts the assumption that $d—c \in H$. So the $(k+1)$th oriented edge is also not in any directed cycle. Now we can get that every directed edge in H_τ is not in any directed cycle. It implies that there are no directed cycles in H_τ, and thus H is a chain graph. ■

Lemma 13 *Let $G^*_{e(V)}$ be the post intervention essential graph with the orientation $e(V)$ and H be the graph constructed by Algorithm 2. We have $G^*_{e(V)} = H$.*

Proof We first prove $G^*_{e(a)} \subseteq H$. We just need to prove that all directed edges in H must be in $G^*_{e(a)}$. We use induction to finish the proof.

After Step 1 of Algorithm 2, all directed edges in H are in $G^*_{e(a)}$. We now prove that the first directed edge oriented at Step 2 of Algorithm 2, such as $b \leftarrow c$, is in $G^*_{e(a)}$. Because $b \leftarrow c$ must be oriented by the rule (i) of Algorithm 2, there must be a node $d \notin \tau$ such that $b—c \leftarrow d$ is the subgraph of H. So $b \leftarrow c \leftarrow d$ must be a subgraph in each $G' \in G^*_{e(a)}$. Otherwise, $b \to c \leftarrow d$ forms a v-structure such that $G' \notin [G]$. Thus we have $b \leftarrow c \in G^*_{e(a)}$.

Suppose that the first k oriented edges at Step 2 of Algorithm 2 are in $G^*_{e(a)}$. We now prove that the $(k+1)$th oriented edge at Step 2 of Algorithm 2 is also in $G^*_{e(a)}$. Denoting the $(k+1)$th oriented edge as $l \leftarrow h$, according to the rules in Algorithm 2, there are two cases to orient $l \leftarrow h$ as shown in Figures 16 and 17.

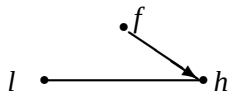


Figure 16: SG_4

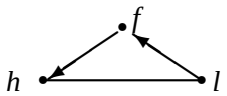


Figure 17: SG_5

In Figure 16, because $f \to h$ is in every DAG $G' \in G^*_{e(a)}$, in order to avoid a new v-structure, we have that $l \leftarrow h$ must be in every DAG $G' \in G^*_{e(a)}$. Thus we have $l \leftarrow h \in G^*_{e(a)}$. In Figure 17, because $l \to f$ and $f \to h$ are in every DAG $G' \in G^*_{e(a)}$, in order to avoid a directed cycle, we have that $h \leftarrow l$ must be in every DAG $G' \in G^*_{e(a)}$. Thus we have $h \leftarrow l \in G^*_{e(a)}$. Now we get that the $(k+1)$th oriented edge at Step 2 of Algorithm 2 is also in $G^*_{e(a)}$. Thus all directed edges in H are also in $G^*_{e(a)}$ and then we have $G^*_{e(a)} \subseteq H$.

Because H is a chain graph by Lemma 12, we also have $H \subseteq G^*$. By Lemma 11, for any undirect edge a—b of H_τ where τ is a chain component of H, there exist G_1 and $G_2 \in G^*_{e(a)}$ such that $a \to b$ occurs in G_1 and $a \leftarrow b$ occurs in G_2. It means that a—b also occurs in $G^*_{e(a)}$. So we have $H \subseteq G^*_{e(a)}$, and then $G^*_{e(a)} = H$. ■

Proof of Theorem 6. By definition of $G^*_{e(V)}$, we have that $G^*_{e(V)}$ has the same skeleton as the essential graph G^* and contains all directed edges of G^*. That is, all directed edges in G^* are also directed in $G^*_{e(V)}$. So property 2 of Theorem 6 holds. Property 3 of Theorem 6 also holds because all DAGs represented by $G^*_{e(V)}$ are Markov equivalent. From Lemmas 12 and 13, we can get that $G^*_{e(V)}$ is a chain graph. ■

Proof of Theorem 7. We first prove property 1. Let $C = ch(V_k) \setminus \tau$. Then $B = ne(V_k) \setminus C$ contains all parents of V_k and the children of V_k in τ. Let $A = An(\{B, V_k\})$ be the ancestor set of all nodes in $\{B, V_k\}$. Since V_i is a parent of V_k for property 1, we have $V_i \in A$. The post-intervention joint distribution of A is

$$P_{V_i}(A) = P'(v_i|pa(v_i)) \prod_{v_j \in A \setminus V_i} P(v_j|pa(v_j)). \tag{1}$$

Let $U = A \setminus \{B, V_k\}$. Then we have from the post-intervention joint distribution (1)

$$\begin{aligned} P_{V_i}(v_k|B) &= \frac{\sum_U P'(v_i|pa(v_i)) \prod_{v_j \in A \setminus V_i} P(v_j|pa(v_j)))}{\sum_{U,V_k} P'(v_i|pa(v_i)) \prod_{V_j \in A \setminus V_i} P(v_j|pa(v_j))} \\ &= \frac{\sum_U P'(v_i|pa(v_i)) \prod_{v_j \in A \setminus \{ch(V_k) \cap \tau, V_k\}} P(v_j|pa(v_j)) \prod_{v_j \in \{ch(V_k) \cap \tau, V_k\}} P(v_j|pa(v_j))}{\sum_{U,V_k} P'(v_i|pa(v_i)) \prod_{v_j \in A \setminus \{ch(V_k) \cap \tau, V_k\}} P(v_j|pa(v_j)) \prod_{v_j \in \{ch(V_k) \cap \tau, V_k\}} P(v_j|pa(v_j))}, \end{aligned}$$

where $\sum_U$ denotes a summation over all variables in the set U.

Below we want to factorize the denominator into a production of summation over U and summation over V_k. First we show that the factor $P'(v_i|pa(v_i)) \prod_{v_j \in A \setminus \{ch(V_k) \cap \tau, V_k\}} P(v_j|pa(v_j))$ does not contain V_k because V_k appears only in the conditional probabilities of $ch(V_k)$ and the conditional probability of V_k. Next we show that $\prod_{v_j \in \{ch(V_k) \cap \tau, V_k\}} P(v_j|pa(v_j))$ does not contain any variable in U. From definition of B, we have $B \supseteq (ch(V_k) \cap \tau)$. Then from definition of U, we have that V_j in $\{ch(V_k) \cap \tau, V_k\}$ is not in U. Now we just need to show that any parent of any node V_j in $\{ch(V_k) \cap \tau, V_k\}$ is also not in U:

1. By definitions of B and U, the parents of V_k is not in U.

2. Consider parents of nodes in $\{ch(V_k) \cap \tau\}$. Let W is such a parent, that is, $W \to V_j$ for $V_j \in \{ch(V_k) \cap \tau\}$. There is a head to head path ($W \to V_j \leftarrow V_k$). We show that W is not in U separately for two cases: $W \in \tau$ and $W \notin \tau$. For the first case of $W \in \tau$, there is an undirected

edge between W and V_k in G^*_τ since there is no v-structure in the subgraph G'_τ for any $G' \in [G]$. Then from definition of B, we have $W \in B$. For the second case of $W \notin \tau$, W must be a parent of V_k by Lemma 10, and then W is in B. Thus we obtain $W \notin U$.

We showed that the factor $\prod_{V_j \in \{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j))$ does not contain any variable in U. Thus the numerator and the summations over U and V_k in the denominator can be factorized as follows

$$\begin{aligned}
&P_{V_i}(v_k|B) \\
&= \frac{\prod_{v_j \in \{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j)) \sum_U P'(v_i|pa(v_i)) \prod_{v_j \in A\setminus\{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j))}{\sum_{V_k} \prod_{v_j \in \{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j)) \sum_U P'(v_i|pa(v_i)) \prod_{v_j \in A\setminus\{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j))} \\
&= \frac{\prod_{v_j \in \{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j))}{\sum_{V_k} \prod_{v_j \in \{ch(V_k)\cap\tau, V_k\}} P(v_j|pa(v_j))} = P(v_k|B).
\end{aligned}$$

Thus we proved property 1.

Property 2 is obvious since manipulating V_i does not change the distribution of its parent V_k. Formally, let $an(V_k)$ be the ancestor set of V_k. If $V_k \in pa(V_i)$, then we have $P_{V_i}(an(v_k), v_k) = P(an(v_k), v_k)$ and thus $P_{V_i}(V_k) = P(V_k)$. ■

Proof of Theorem 8. Manipulating a node V_i will orient all of undirected edges connecting V_i. Thus the orientations of undirected edges do not depend on the order in which the variables are manipulated. If a sequence $\mathcal{S}$ is sufficient, then its permutation is also sufficient. ■

Proof of Theorem 9. Suppose that $\mathcal{S} = (V_1, \ldots, V_K)$ is a sufficient set. We delete a node, say V_i, from $\mathcal{S}$, and define $\mathcal{S}'_{[i]} = \mathcal{S} \setminus \{V_i\}$. If the set $\mathcal{S}'_{[i]}$ is no longer sufficient, then we can add other variables to $\mathcal{S}'_{[i]}$ without adding V_i such that $\mathcal{S}'_{[i]}$ becomes to be sufficient. This is feasible since any undirected edge can be oriented by manipulating either of its two nodes. Thus we have $\bigcap_{i=1}^{K} \mathcal{S}'_{[i]} = \varnothing$. Since all $\mathcal{S}'_{[i]}$ belong to $\mathbb{S}$, we proved $\bigcap_{\mathcal{S}\in\mathbb{S}} \mathcal{S} = \varnothing$.

Similarly, for each minimum sequence $\mathcal{S}$, we can define $\mathcal{S}'_{[i]}$ such that it does not contain V_i and it is a minimum sufficient set. Thus the intersection of all minimum sufficient sets is empty. ■

References

C. Aliferis, I. Tsamardinos, A. Statnikov and L. Brown. Causal explorer: A probabilistic network learning toolkkit for biomedical discovery. In *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, pages 371-376, 2003.

S. A. Andersson, D. Madigan and M. D. Perlman. A characterization of markov equivalence classes for acyclic digraphs. *Annals of Statistics*, 25:505-541, 1997.

R. Castelo and M. D. Perlman. Learning Essential graph Markov models from data. In *Proceedings 1st European Workshop on Probabilistic Graphical Models*, pages 17-24, 2002.

G. F. Cooper and C. Yoo. Causal discovery from a mixture of experimental and observational data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 116-125, 1999.

N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303(5659):799-805, 2004.

Y. He, Z. Geng and X. Liang. Learning causal structures based on Markov equivalence class. In *ALT, Lecture Notes in Artificial Intelligence* 3734, pages 92-106, 2005.

D. Heckerman, D. Geiger and D. M. Chickering. Learning Bayesian networks: The Combination of knowledge and statistical data. *Machine Learning*, 20:197-243, 1995.

D. Heckerman. A Bayesian approach to causal discovery. *Data Mining and Knowledge Discovery*, 1(1):79-119, 1997.

R. Jansen, H. Y. Yu and D. Greenbaum. A Bayesian networks approach for predicting protein-protein interactions from genomic data. *Science*, 302(5644):449-453, 2003.

S. L. Lauritzen. *Graphical Models*. Oxford Univ. Press. 1996.

S. L. Lauritzen, T. S. Richardson. Chain graph models and their casual interpretations. *Journal of the Royal Statistical society series B-statistical methodology*,64:321-348, Part 3, 2002.

M. Kalisch, P. Buhlmann. Estimating high-dimensional directed acyclic graphs with the PC-algorithm.*Journal of Machine Learning Research* 8, 613-636, 2007.

K. P. Murphy. Active Learning of Causal Bayes Net Structure, *Technical Report*, Department of Computer Science, University of California Berkeley, 2001.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

J. Pearl. Graphical models, causality and intervention. *Statist. Sci.*, 8:266-269, 1993.

J. Pearl. Causal inference from indirect experiments. *Artifcal Intelligence in Medicine*, 7:561-582, 1995.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

P. Spirtes, C. Glymour, R. Scheines. *Causation, Prediction, and Search*. MIT Press, Cambridge, MA, second edition, 2000.

J. Tian and J. Pearl. Causal Discovery from Changes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 512-521, 2001a.

J. Tian and J. Pearl. Causal Discovery from Changes: a Bayesian Approach, UCLA Cognitive Systems Laboratory, Technical Report (R-285), 2001b.

S. Tong and D. Koller. Active learning for structure in bayesian networks. In *International Joint Conference on Artificial Intelligence*, pages 863-869, 2001.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 220-227, 1990.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley, New York. 1990.

Journal of Machine Learning Research 9 (2008) 2549-2578 Submitted 10/07; Published 11/08

Stationary Features and Cat Detection

François Fleuret FLEURET@IDIAP.CH
IDIAP Research Institute,
Centre du Parc, Rue Marconi 19,
Case Postale 592,
1920 Martigny, Switzerland

Donald Geman GEMAN@JHU.EDU
Johns Hopkins University,
Clark Hall 302A,
3400 N. Charles Street
Baltimore, MD 21218, USA

Editor: Pietro Perona

Abstract

Most discriminative techniques for detecting instances from object categories in still images consist of looping over a partition of a pose space with dedicated binary classifiers. The efficiency of this strategy for a complex pose, that is, for fine-grained descriptions, can be assessed by measuring the effect of sample size and pose resolution on accuracy and computation. Two conclusions emerge: (1) fragmenting the training data, which is inevitable in dealing with high in-class variation, severely reduces accuracy; (2) the computational cost at high resolution is prohibitive due to visiting a massive pose partition.

To overcome data-fragmentation we propose a novel framework centered on pose-indexed features which assign a response to a pair consisting of an image and a pose, and are designed to be stationary: the probability distribution of the response is always the same if an object is actually present. Such features allow for efficient, one-shot learning of pose-specific classifiers. To avoid expensive scene processing, we arrange these classifiers in a hierarchy based on nested partitions of the pose; as in previous work on coarse-to-fine search, this allows for efficient processing.

The hierarchy is then ”folded” for training: all the classifiers at each level are derived from one base predictor learned from all the data. The hierarchy is ”unfolded” for testing: parsing a scene amounts to examining increasingly finer object descriptions only when there is sufficient evidence for coarser ones. In this way, the detection results are equivalent to an exhaustive search at high resolution. We illustrate these ideas by detecting and localizing cats in highly cluttered greyscale scenes.

Keywords: supervised learning, computer vision, image interpretation, cats, stationary features, hierarchical search

1. Introduction

This work is about a new strategy for supervised learning designed for detecting and describing instances from semantic object classes in still images. Conventional examples include faces, cars and pedestrians. We want to do more than say whether or not there are objects in the scene; we want to provide a description of the pose of each detected instance, for example the locations of certain landmarks. More generally, pose could refer to any properties of object instantiations which

are not directly observed; however, we shall concentrate on geometric descriptors such as scales, orientations and locations.

The discriminative approach to object detection is to induce classifiers directly from training data without a data model. Generally, one learns a pose-specific binary classifier and applies it many times (Rowley et al., 1998; Papageorgiou and Poggio, 2000; Viola and Jones, 2004; LeCun et al., 2004). Usually, there is an outer loop which visits certain locations and scales with a sliding window, and a purely learning-based module which accommodates all other sources of variation and predicts whether or not a sub-window corresponds to a target. Parsing the scene in this manner already exploits knowledge about transformations which preserve object identities. In particular, translating and scaling the training images to a reference pose allows for learning a base classifier with all the training examples. We refer to such learning methods, which use whole image transforms in order to normalize the pose, as "data-aggregation" strategies.

However such transforms, which must be applied online during scene parsing as well as offline during training, may be costly, or even ill-defined, for complex poses. How does one "normalize" the pose of a cat? In such cases, an alternative strategy, which we call "data-fragmentation," is to reduce variation by learning many separate classifiers, each dedicated to a sub-population of objects with highly constrained poses and each trained with only those samples satisfying the constraints. Unfortunately, this approach to invariance might require a massive amount of training data due to partitioning the data. As a result, the discriminative approach has been applied almost exclusively to learning rather coarse geometric descriptions, such as a facial landmark and in-plane orientation, by some form of data-aggregation. Summarizing: aggregating the data avoids sparse training but at the expense of costly image transforms and restrictions on the pose; fragmenting the data can, in principle, accommodate a complex pose but at the expense of crippling performance due to impoverished training.

A related trade-off is the one between computation and pose resolution. Sample size permitting, a finer subpopulation (i.e., higher pose resolution) allows for training a more discriminating classifier. However, the more refined the pose partitioning, the more online computation because regardless of how the classifiers are trained, having more of them means more costly scene parsing. This trade-off is clearly seen for cascades (Viola and Jones, 2004; Wu et al., 2008): at a high true positive rate, reducing false positives could only come at the expense of considerable computation due to dedicating the cascade to a highly constrained pose, hence increasing dramatically the number of classifiers to train and evaluate in order to parse the scene.

To set the stage for our main contribution, a multi-resolution framework, we attempted to quantify these trade-offs with a single-resolution experiment on cat detection. We considered multiple partitions of the space of poses at different resolutions or granularities. For each partition, we built a binary classifier for each cell. There are two experimental variables besides the resolution of the partition: the data may be either fragmented or aggregated during training and the overall cost of executing all the classifiers may or may not be equalized. Not surprisingly, the best performance occurs with aggregated training at high resolution, but the on-line computational cost is formidable. The experiment is summarized in an Appendix A and described in detail in Fleuret and Geman (2007).

Our framework is designed to avoid these trade-offs. It rests on two core ideas. One, which is not new, is to control online computation by using a hierarchy of classifiers corresponding to a recursive partitioning of the pose space, that is, parameterizations of increasing complexity. A richer parametrization is considered only when "necessary", meaning the object hypothesis cannot

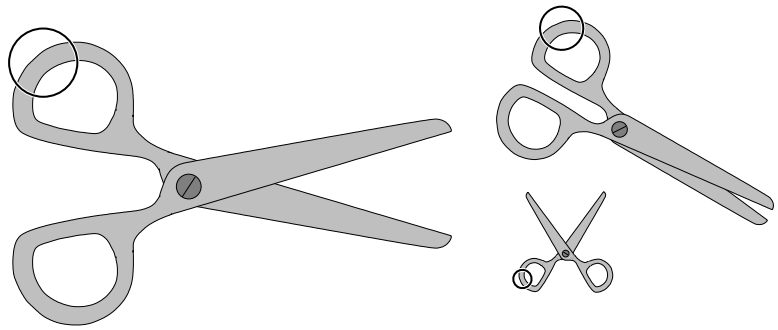


Figure 1: An idealized example of stationary features. The pose of the scissors could be the locations of the screw and the two tips, in which case one might measure the relative frequency a particular edge orientation inside in a disc whose radius and location, as well as the chosen orientation, depends on the pose. If properly designed, the response statistics have a distribution which is invariant to the pose when in fact a pair of scissors is present (see § 3.3).

be ruled out with a simpler one (see, e.g., Fleuret and Geman, 2001; Stenger et al., 2006). (Note that cascades are efficient for a similar reason - they are coarse-to-fine in terms of background rejection.) However, hierarchical organization alone is unsatisfactory because it does not solve the data-fragmentation problem. Unless data can be synthesized to generate many dedicated sets of positive samples, one set per node in the hierarchy, the necessity of training a classifier for every node leads to massive data fragmentation, hence small node-specific training sets, which degrades performance.

The second idea, the new one, is to avoid data-fragmentation by using pose-specific classifiers trained with "stationary features", a generalization of the underlying implicit parametrization of the features by a scale and a location in all the discriminative learning techniques mentioned earlier. Each stationary feature is "pose-indexed" in the sense of assigning a numerical value to each combination of an image and a pose (or subset of poses). The desired form of stationarity is that, for any given pose, the *distribution* of the responses of the features over images containing an object at that pose does not depend on the pose. Said another way, if an image and an object instance at a given pose are selected, and only the responses of the stationary features are provided, one cannot guess the pose. This is illustrated in Figure 1: knowing only the proportion of edges at a pose-dependent orientation in the indicated disk provides no information about the pose of the scissors.

Given that objects are present, a stationary feature evaluated at one pose is then the "same" as at any other, but not in a literal, point-wise sense as functions, but rather in the statistical, population sense described above. In particular, stationary features are not "object invariants" in the deterministic sense of earlier work (Mundy and Zisserman, 1992) aimed at discovering algebraic and geometric image functionals whose actual values were invariant with respect to the object pose. Our aim is less ambitious: our features are only "invariant" in a statistical sense. But this is enough to use all the data to train each classifier.

Of course the general idea of connecting features with object poses is relatively common in object recognition. As we have said, pose-indexing is done implicitly when transforming images to a

reference location or scale, and explicitly when translating and scaling Haar wavelets or edge detectors to compute the response of a classifier for a given location and scale. Surprisingly, however, this has not been formulated and analyzed in general terms, even though stationarity is all that is needed to aggregate data while maintaining the standard properties of a training set. Stationarity makes it possible, and effective, to analytically construct an entire family of pose-specific classifiers—all those at a given level of the hierarchy—using one base classifier induced from the entire training set. In effect, each pose-specific classifier is a "deformation" of the base classifier. Hence the number of classifiers to train grows linearly, not exponentially, with the depth of the pose hierarchy. This is what we call a folded hierarchy of classifiers: a tree-structured hierarchy is collapsed, like a fan, into a single chain for training and then expanded for coarse-to-fine search.

The general formulation opens the way for going beyond translation and scale, for example for training classifiers based on checking consistency among parts or deformations of parts instead of relying exclusively on their marginal appearance. Such a capability is indeed exploited by the detector we designed for finding cats and greatly improves the performance compared to individual part detection. This gain is shown in Figure 2, the main result of the paper, which compares ROC curves for two detectors, referred to as "H+B" and "HB" in the figure. In the "H+B" case, two separate detectors are trained by data aggregation, one dedicated to heads and the other to bodies; the ROC curve is the best we could do in combining the results. The "HB" detector is a coordinated search based on stationary features and a two-level hierarchy; the search for the belly location in the second-level is conditional on a pending head location and data fragmentation is avoided with pose-indexed features in a head-belly frame. A complete explanation appears in § 6.

In §2, we summarize previous, related work on object detection in still images. Our notation and basic ideas are formally introduced in §3, highlighting the difference between transforming the signal and the features. The motivational experiment, in which we substantiate our claims about the forced trade-offs when conventional approaches are applied to estimating a complex pose, could be read at this point; see Appendix A. Embedding pose-indexed classifiers in a hierarchy is described in §4 and the base classifier, a variation on boosting, is described in §5. In §6 we present our main experiment - an application of the entire framework, including the specific base features, pose hierarchy and pose-indexed features, to detecting cats in still images. Finally, some concluding remarks appear in §7.

2. Related Work

We characterize other work in relation to the two basic components of our detection strategy: explicit modeling of a hidden pose parameter, as in many generative and discriminative methods, and formulating detection as a controlled "process of discovery" during which computation is invested in a highly adaptive and unbalanced way depending on the ambiguities in the data.

2.1 Hidden Variables

A principal source of the enormous variation in high-dimensional signals (e.g., natural images) is the existence of a hidden state which influences many components (e.g., pixel intensities) simultaneously, creating complex statistical dependencies among them. Still, even if this hidden state is of high dimension, it far simpler than the observable signal itself. Moreover, since our objective is to interpret the signal at a semantic level, much of the variation in the signal is irrelevant.

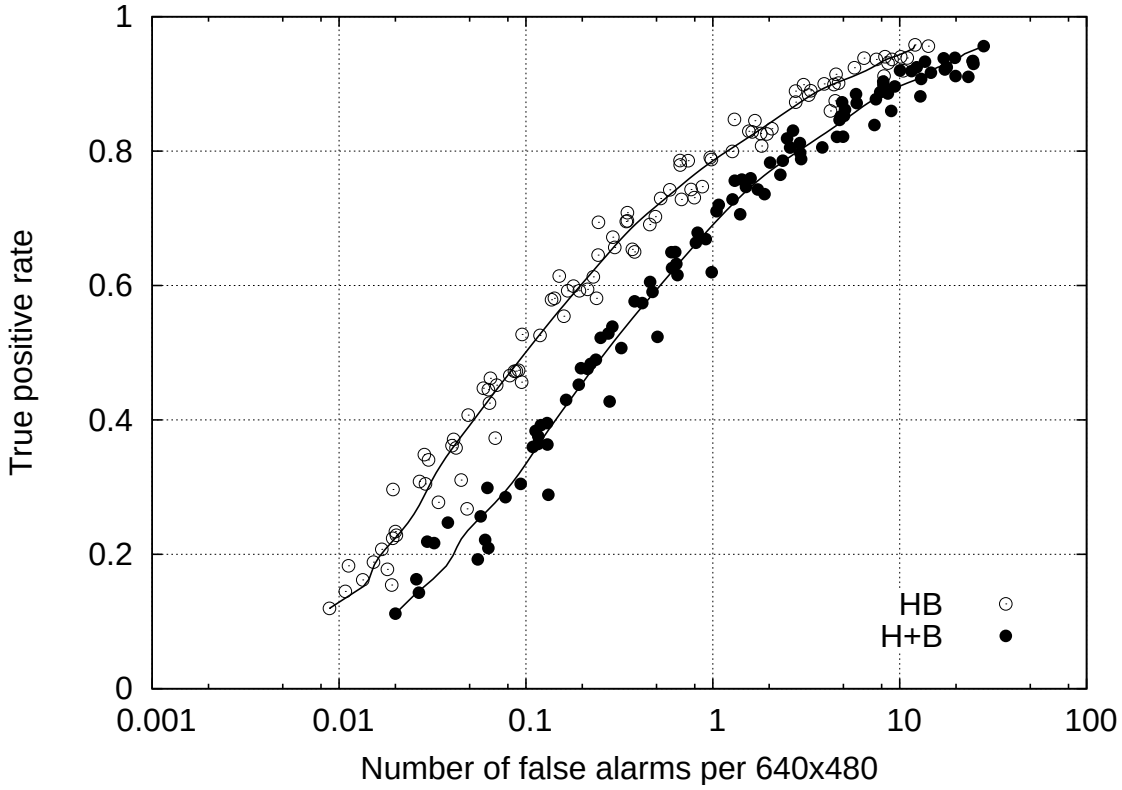


Figure 2: ROC curves for head-belly detection. The criterion for a true detection is that the estimates of the head location, head size and belly location all be close to the true pose (see § 6.6). The H+B detector is built from separate head and body detectors while the HB detector is built upon pose indexed features (see § 6.5).

In fact, conditioning on the value of the hidden state, which means, in practice, testing for the presence of a target with a given pose, often leads to very simple, yet powerful, statistical models by exploiting the increased degree of independence among the components of the signal. This means decisions about semantic content can be based on directly aggregating evidence (naive Bayes). The problem is computational: there are many possible hidden states.

The extreme application of this conditioning paradigm is classical template matching (Grenander, 1993): if the pose is rich enough to account for all non-trivial statistical variation, then even a relatively simple metric can capture the remaining uncertainty, which is basically noise. But this requires intense online computation to deform images or templates many times. One motivation of our approach is to avoid such online, global image transformations.

Similarly, the purest learning techniques, such as boosting (Viola and Jones, 2004) and convolution neural networks (LeCun et al., 2004), rely on explicitly searching through a subset of possible scales and locations in the image plane; that is, coarse scale and coarse location are not learned. Nor is invariance to illumination, usually handled at the feature level. However, invariance to other

geometric aspects of the pose, such as rotation, and to fine changes in scale and translation, are accommodated implicitly, that is, during classifier training.

On the contrary, "Part and Structure" models and other generative (model-based) approaches aim at more complex representations in terms of properties of "parts" (Li et al., 2003; Schneiderman and Kanade, 2004; Crandall and Huttenlocher, 2006). However, tractable learning and computation often require strong assumptions, such as conditional independence in appearance and location. In some cases, each part is characterized by the response of a feature detector, and the structure itself—the arrangement of parts—can either be captured by a complex statistical model, incurring severe computation in both training and testing, or by a simple model by assuming conditional independence among part locations given several landmarks, which can lead to very efficient scene parsing with the use of distance transforms. Some of these techniques do extend to highly articulated and deformable objects; see, for example, Huttenlocher and Felzenszwalb (2005). Still, modeling parts of cats (heads, ears, paws, tails, etc.) in this framework may be difficult due to the low resolution and high variation in their appearance, and in the spatial arrangements among them. Compositional models (Geman et al., 2002; Zhu and Mumford, 2006; Ommer et al., 2006) appear promising. Among these, in the "patchwork of parts" model (citepamit-trouve2007, the feature extractors are, like here, defined with respect to the pose of the object to detect, in that case a series of control points. This strategy allows for aggregating training samples with various poses through the estimation of common distributions of feature responses.

2.2 A Process of Discovery

We do not regard the hidden pose as a "nuisance" parameter, secondary to detection itself, but rather as part of what it means to "recognize" an object. In this regard, we share the view expressed in Geman et al. (2002), Crandall and Huttenlocher (2006) and elsewhere that scene interpretation should go well beyond pure classification towards rich annotations of the instantiations of the individual objects detected.

In particular, we envision detection as an organized process of discovery, as in Amit et al. (1998), and we believe that computation is a crucial issue and should be highly concentrated. Hierarchical techniques, which can accomplish focusing, are based on a recursive partitioning of the pose space (or object/pose space), which can be either ad-hoc (Geman et al., 1995; Fleuret and Geman, 2001) or learned (Stenger et al., 2006; Gangaputra and Geman, 2006). There is usually a hierarchy of classifiers, each one trained on a dedicated set of examples—those carrying a pose in the corresponding cell of the hierarchy. Often, in order to have enough data to train the classifiers, samples must be generated synthetically, which requires a sophisticated generative model.

Our work is also related to early work on hierarchical template-matching (Gavrila, 1998) and hierarchical search of pose space using branch and bound algorithms (Huttenlocher and Rucklidge, 1993), and to the cascade of classifiers in Viola and Jones (2004) and Wu et al. (2008).

Relative to the tree-based methods, we use the stationary features to aggregate data and build only one base classifier per level in the hierarchy, from which all other classifiers are defined analytically. Finally, the fully hierarchical approach avoids the dilemma of cascades, namely the sacrifice of selectivity if the pose space is coarsely explored and the sacrifice of computation if it is finely explored, that is, the cascades are dedicated to a very fine subset of poses.

3. Stationary Features

We regard the image as a random variable I assuming values in $\mathcal{I}$. The set of possible poses for an object appearing in I is $\mathcal{Y}$. We only consider geometric aspects of pose, such as the sizes of well-defined parts and the locations of distinguished points.

Let $\mathcal{Y}_1,\ldots,\mathcal{Y}_K$ be a partition of $\mathcal{Y}$. As we will see in § 4, we are interested in partitions of varying granularities for the global process of detection, ranging from rather coarse resolution (small K) to rather fine resolution (larger K), but in this section we consider one fixed partition.

For every $k = 1\ldots K$, let Y_k be a Boolean random variable indicating whether or not there is a target in I with pose in $\mathcal{Y}_k$. The binary vector $(Y_1,\ldots,Y_K)$ is denoted $\mathbf{Y}$.

In the case of merely detecting and localizing an object of fixed size in a gray-scale image of size $W \times H$, natural choices would be $\mathcal{I} = [0,1]^{WH}$ and $\mathcal{Y} = [0,W] \times [0,H]$, the image plane itself; that is, the pose reduces to one location. If the desired detection accuracy were 5 pixels, then the pose cells might be disjoint 5×5 blocks and K would be approximately $\frac{WH}{25}$. On the other hand, if the pose accommodated scale and multiple points of interest, then obviously the same accuracy in the prediction would lead to a far larger K, and any detection algorithm based on looping over pose cells would be highly costly.

We denote by $\mathcal{T}$ a training set of images labeled with the presences of targets

$$\mathcal{T} = \left\{ \left(I^{(t)}, \mathbf{Y}^{(t)} \right) \right\}_{1 \le t \le T},$$

where each $I^{(t)}$ is a full image, and $\mathbf{Y}^{(t)}$ is the Boolean vector indicating the pose cells occupied by targets in $I^{(t)}$. We write

$$\xi : \mathcal{I} \to \mathbb{R}^N,$$

for a family of N image features such as edge detectors, color histograms, Haar wavelets, etc. These are the "base features" $(\xi_1,\ldots,\xi_N)$ which will be used to generate our stationary feature vector. We will write $\xi(I)$ when we wish to emphasize the mapping and just ξ for the associated random variable. The dimension N is sufficiently large to account for all the variations of the feature parameters, such as locations of the receptive fields, orientations and scales of edges, etc.

In the next section, § 3.1, we consider the problem of "data-fragmentation", meaning that specialized predictors are trained with subsets of the positive samples. Then, in § 3.2, we formalize how fragmentation has been conventionally avoided in simple cases by normalizing the signal itself; we then propose in § 3.3 the idea of pose-indexed, stationary features, which avoids global signal normalization both offline and online and opens the way for dealing with complex pose spaces.

3.1 Data Fragmentation

Without additional knowledge about the relation between $\mathbf{Y}$ and I, the natural way to predict Y_k for each $k = 1\ldots K$ is to train a dedicated classifier

$$f_k : \mathcal{I} \to \{0,1\}$$

with the training set

$$\left\{ \left(I^{(t)}, Y_k^{(t)} \right) \right\}_{1 \le t \le T}$$

derived from $\mathcal{T}$. This corresponds to generating a single sample from each training scene, labeled according to whether or not there is a target with pose in $\mathcal{Y}_k$. This is *data-fragmentation*: training

$\mathcal{Y}$, the pose space
$\mathcal{Y}_1,\ldots,\mathcal{Y}_K$, a partition of the pose space $\mathcal{Y}$
$\mathcal{Z}$, a $W \times H$ pixel lattice
$I = \{0,\ldots,255\}^{\mathcal{Z}}$, a set of gray-scale images of size $W \times H$
I, a random variable taking values in I
Y_k, a Boolean random variable indicating if there is a target in I with pose in $\mathcal{Y}_k$
$\mathbf{Y} = (Y_1,\ldots,Y_K)$
T, the number of training images, each with or without targets
$\mathcal{T} = \left\{\left(I^{(t)},\mathbf{Y}^{(t)}\right)\right\}_{1\leq t\leq T}$, the training set
$f_k : I \to \{0,1\}$, a predictor of Y_k based on the image
Q, number of image features
$\xi : I \to \mathbb{R}^N$, a family of base image features
$\psi : \{1,\ldots,K\} \times I \to I$, an image transformation intended to normalize a given pose
$\mathbf{X} : \{1,\ldots,K\} \times I \to \mathbb{R}^Q$, a family of pose-indexed features
$\mathbf{X}(k)$, the r.v. corresponding to $\mathbf{X}(k,I)$
$g : \mathbb{R}^Q \to \{0,1\}$, a predictor trained from all the data

Table 1: Notation

f_k involves only those data which exactly satisfy the pose constraint; no synthesis or transformations are exploited to augment the number of samples available for training. Clearly, the finer the partitioning of the pose space $\mathcal{Y}$, the fewer positive data points are available for training each f_k.

Such a strategy is evidently foolhardy in the standard detection problems where the pose to be estimated is the location and scale of the target since it would mean separately training a predictor for every location and every scale, using as positive samples only full scenes showing an object at that location and scale. The relation between the signal and the pose is obvious and normalizing the positive samples to a common reference pose by translating and scaling them is the natural procedure; only one classifier is trained with all the data. However, consider a face detection task for which the faces to detect are known to be centered and of fixed scale, but are of unknown out-of-plane orientation. Unless 3D models are available, from which various views can be synthesized, the only course of action is data-fragmentation: partition the pose space into several cells corresponding to different orientation ranges and train a dedicated, range-specific classifier with the corresponding positive samples.

3.2 Transforming the Signal to Normalize the Pose

As noted above, in simple cases the image samples can be normalized in pose. More precisely, both training and scene processing involve normalizing the image through a pose-indexed transformation

$$\psi : \{1,\ldots,K\} \times I \to I.$$

The "normalization property" we desire with respect to ξ is that the conditional probability distribution of $\xi(\psi(k,I))$ given $Y_k = 1$ be the same for every $1 \leq k \leq K$.

The intuition behind this property is straightforward. Consider for instance a family of edge detectors and consider again a pose consisting of a single location z. In such a case, the transformation ψ applies a translation to the image to move the center of pose cell $\mathcal{Y}_k$ to a reference location. If

a target was present with a pose in $\mathcal{Y}_k$ in the original image, it is now at a reference location in the transformed image, and the distribution of the response of the edge detectors in that transformed image does not depend on the initial pose cell $\mathcal{Y}_k$.

We can then define a new training set

$$\left\{\left(\xi\left(\psi(k,I^{(t)})\right),Y_k^{(t)}\right)\right\}_{1\le k\le K,1\le t\le T}$$

with elements residing in $\mathbb{R}^N\times\{0,1\}$. Due to the normalization property, and under mild conditions, the new training set indeed consists of independent and identically distributed components (see the discussion in the following section). Consequently, this set allows for training a classifier

$$g:\mathbb{R}^N\to\{0,1\}$$

from which we can analytically define a predictor of Y_k for any k by

$$f_k(I)=g\left(\xi(\psi(k,I))\right).$$

This can be summarized algorithmically as follows: In order to predict if there is a target in image I with pose in $\mathcal{Y}_k$, first normalize the image with ψ so that a target with pose in $\mathcal{Y}_k$ would be moved to a reference pose cell, then extract features in that transformed image using ξ, and finally evaluate the response of the predictor g from the computed features.

3.3 Stationary Features

The pose-indexed, image-to-image mapping ψ is computationally intensive for any non-trivial transformation. Even rotation or scaling induces a computational cost of $O(WH)$ for every angle or scale to test during scene processing, although effective shortcuts are often employed. Moreover, this transformation does not exist in the general case. Consider the two instances of cats shown in Figure 3. Rotating the image does not allow for normalizing the body orientation without changing the head orientation, and designing a non-affine transformation to do so would be unlikely to produce a realistic cat image as well as be computationally intractable when done many times. Finally, due to occlusion and other factors, there is no general reason *a priori* for ψ to even exist.

Instead, we propose a different mechanism for data-aggregation based on pose-indexed features which directly assign a response to a pair consisting of an image and a pose cell and which satisfy a stationarity requirement. This avoids assuming the existence of a normalizing mapping in the image space, not to mention executing such a mapping many times online.

A **stationary feature vector** *is a pose-indexed mapping*

$$\mathbf{X}:\{1,\ldots,K\}\times I\to\mathbb{R}^Q,$$

with the property that the probability distribution

$$P(\mathbf{X}(k)=\mathbf{x}\,|\,Y_k=1),\ \mathbf{x}\in\mathbb{R}^Q \tag{1}$$

is the same for every $k=1,\ldots,K$, where $\mathbf{X}(k)$ denotes the random variable $\mathbf{X}(k,I)$.

The idea can be illustrated with two simple examples, a pictorial one in Figure 1 and a numerical one in § 3.4.

Figure 3: Aggregating data for efficient training by normalizing the pose at the image level is difficult for complex poses. For example, linear transformations cannot normalize the orientation of the body without changing that of the head.

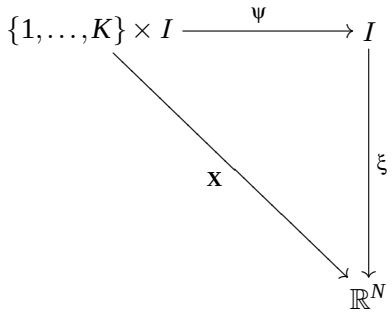
In practice, the relationship with ξ, the base feature vector, is simply that the components of the feature vector $\mathbf{X}(k)$ are chosen from among the components of ξ; the choice depends on k. In this case, we can write

$$\mathbf{X}(k) = (\xi_{\pi_1(k)}, \xi_{\pi_2(k)}, \ldots, \xi_{\pi_Q(k)}),$$

where $\{\pi_1(k), \ldots, \pi_Q(k)\} \subset \{1, \ldots, N\}$ is the ordered selection for index k. The ordering matters because we want (1) to hold and hence there is a correspondence among individual components of $\mathbf{X}(k)$ from one pose cell to another.

Note: We shall refer to (1) as the "stationarity" or "weak invariance" assumption. As seen below, this property justifies data-aggregation in the sense of yielding an aggregated training set satisfying the usual conditions. Needless to say, however, demanding that this property be satisfied exactly is not practical, even arguably impossible. In particular, with our base features, various discretizing effects come into play, including using quantized edge orientations and indexing base features with rectangular windows. Even designing the pose-indexed features to approximate stationarity by appropriately selecting and ordering the base features is non-trivial; indeed, it is the main challenge in our framework. Still, using pose-indexed features which are even approximately stationary will turn out to be very effective in our experiments with cat detection.

The contrast between signal and feature transformations can be illustrated with the following commutative diagram: Instead of first applying a normalizing mapping ψ to transform I in accordance with a pose cell k, and then evaluating the base features, we directly compute the feature responses as functions of both the image and the pose cell.



Once provided with $\mathbf{X}$, a natural training set consisting of TK samples is provided by

$$\mathcal{T}_{agg} = \left\{ \left(\mathbf{X}^{(t)}(k), Y_k^{(t)} \right) \right\}_{1 \le t \le T, 1 \le k \le K}. \tag{2}$$

Under certain conditions, the elements of this training set will satisfy the standard assumption of being independent and identically distributed. One condition, the key one, is stationarity, but technically three additional conditions would be required: 1) property (1) extend to conditioning on $Y_k = 0$; 2) the "prior" distribution $P(Y_k = 1)$ be the same for every $k = 1,\ldots,K$; 3) for each t, the samples $\mathbf{X}^{(t)}(k), k = 1,\ldots,K$, be independent. The first condition says that the background distribution of the pose-indexed features is spatially homogeneous, the second that all pose cells are *a priori* equally likely and the third, dubious but standard, says that the image data associated with different pose cells are independent despite some overlap. In practice, we view these as rough guidelines; in particular, we make no attempt to formally verify any of them.

It therefore makes sense to train a predictor $g : \mathbb{R}^Q \to \{0,1\}$ using the training set (2). We can then *define*

$$f_k(I) = g(\mathbf{X}(k,I)),\ k = 1,\ldots,K.$$

Notice that the family of classifiers $\{f_k\}$ is also "stationary" in the sense that conditional distribution of f_k given $Y_k = 1$ does not depend on k.

3.4 Toy Example

We can illustrate the idea of stationary features with a very simple roughly piecewise constant, one-dimensional signal $I(n), n = 1,...,N$. The base features are just the components of the signal itself: $\xi(I) = I$. The pose space is

$$\mathcal{Y} = \left\{ (\theta_1, \theta_2) \in \{1,\ldots,N\}^2, 1 < \theta_1 < \theta_2 < N \right\}$$

and the partition is the finest one whose cells are individual poses $\{(\theta_1,\theta_2)\}$; hence $K = |\mathcal{Y}|$. For simplicity, assume there is at most one object instance, so we can just write $Y = (\theta_1,\theta_2) \in \mathcal{Y}$ to denote an instance with pose (θ_1,θ_2). For $\mathbf{u} = (u_1,...,u_N) \in \mathbb{R}^N$, the conditional distribution of I given Y is

$$\begin{aligned} P(I = \mathbf{u} \,|\, Y = (\theta_1,\theta_2)) &= \prod_n P(I(n) = u_n \,|\, Y = (\theta_1,\theta_2)) \\ &= \prod_{n<\theta_1} \phi_0(u_n) \prod_{\theta_1 \le n \le \theta_2} \phi_1(u_n) \prod_{\theta_2 < n} \phi_0(u_n) \end{aligned}$$

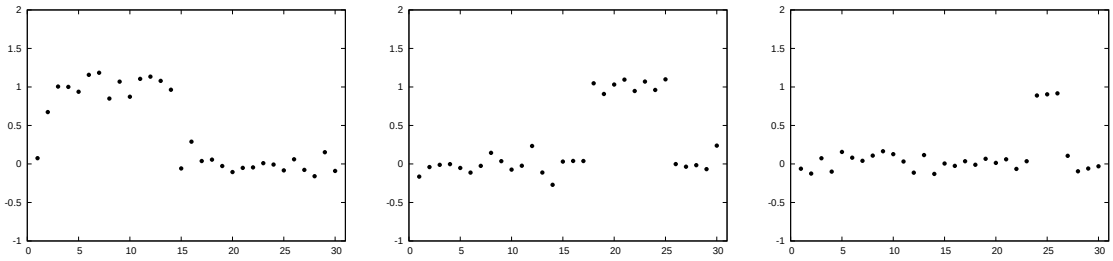


Figure 4: Examples of toy scenes

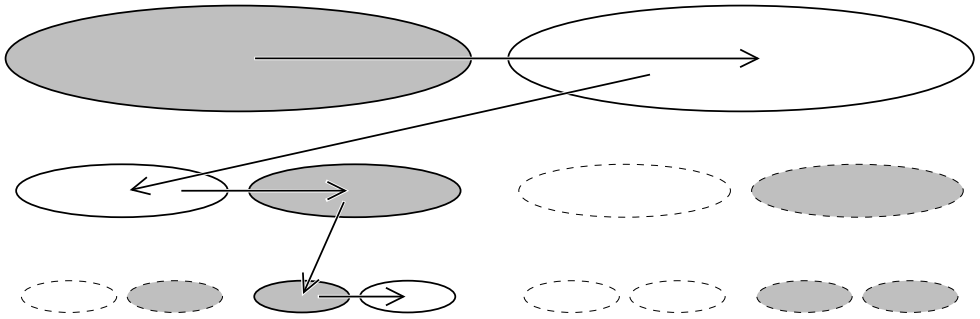


Figure 5: Hierarchical detection. Each ellipse on stands for a pose cell $\mathcal{Y}_k^{(d)}, k = 1,\ldots,K_d, d = 1,\ldots,D$. Here, $D = 3$ and $K_1 = 2$, $K_2 = 4$, $K_3 = 8$. Gray ellipses correspond to pose cells whose $f_k^{(d)}$ respond positively, and dashed ellipses correspond to pose cells whose classifiers are not evaluated during detection. As shown by the arrows, the algorithm ignores all sub-cells of a cell whose classifier responds negatively.

where ϕ_μ is a normal law with mean μ and standard deviation 0.1. Hence the signal fluctuates around 0 on the "background" and around 1 on the target, see Figure 4.

We define a four-dimensional pose-indexed feature vector taking the values of the signal at the extremities of the target, that is

$$\mathbf{X}((\theta_1,\theta_2),I) = (I(\theta_1-1),I(\theta_1),I(\theta_2),I(\theta_2+1)).$$

Clearly,

$$P(\mathbf{X}(\theta_1,\theta_2) = (x_1,x_2,x_3,x_4)\,|\,Y_{\theta_1,\theta_2} = 1) = \phi_0(x_1)\phi_1(x_2)\phi_1(x_3)\phi_0(x_4)$$

which is not a function of θ_1,θ_2. Consequently, $\mathbf{X}$ is stationary and the common law in (1) is $\phi_0 \times \phi_1 \times \phi_1 \times \phi_0$.

4. Folded Hierarchies

We have proposed normalizing the samples through a family of pose-indexed features instead of whole image transforms in order to avoid fragmentation of the data. Since only one classifier must be built for any partition of the pose space, and no longer for every cell of such a partition, neither

the cost of learning nor the required size of the training set grows linearly with the number K of pose cells in the partition. However, one main drawback remains: We must still visit all the pose cells online, which makes the cost of scene processing itself linear in K.

A natural strategy to address computational cost is an hierarchical search strategy based upon a recursive partitioning of $\mathcal{Y}$. As in previous work (Fleuret and Geman, 2001; Gangaputra and Geman, 2006), there is a succession of nested partitions of increasing resolution and a binary classifier assigned to each cell. Given such a hierarchy, the detection process is adaptive: a classifier is evaluated for a certain pose cell only if all the classifiers for its ancestor cells have been evaluated and responded positively.

Note: This is *not* a decision tree, both in terms of representation and processing. The hierarchy recursively partitions the space of hidden variables not the feature space, and the edges from a node to its children do not represent the possible values of a node classifier. Moreover, during processing, a data point may traverse many branches at once and may reach no leaves or reach many leaves.

The crucial difference with previous work is that, using stationary features, only one classifier must be trained for each level, not one classifier for each cell. In essence, the hierarchy is "folded" (like a fan) for training: The entire learning strategy described in §3 is repeated for each level in the hierarchy. This is quite straightforward and only summarized below.

Consider a sequence of partitions of $\mathcal{Y}$

$$\left\{\mathcal{Y}_1^{(d)},\ldots,\mathcal{Y}_{K_d}^{(d)}\right\},\quad 1\leq d\leq D,$$

for which any cell $\mathcal{Y}_k^d$ for $k=1,\ldots,K_{d+1}$, is a (disjoint) union of cells at the next level $d+1$. Consequently, we can identify every $\mathcal{Y}_k^{(d)}$ with the node of a multi-rooted tree: A leaf node for $d=D$ and an internal node otherwise. A three-level hierarchy is shown in Figure 5.

Given such a pose hierarchy, we can construct a scene parsing algorithm aimed at detecting all instances of objects at a pose resolution corresponding to the finest partition. Again, the processing strategy is now well-known. This algorithm has the desirable property of concentrating computation on the ambiguous pose-image pairs.

Let $Y_k^{(d)}$ denote a Boolean random variable indicating whether or not there is a target in I with pose in $\mathcal{Y}_k^{(d)}$ and let $\mathbf{X}^{(d)}$ denote a pose-indexed feature vector adapted to the partition $\{\mathcal{Y}_1^{(d)},\ldots,\mathcal{Y}_{K_d}^{(d)}\}$. For each level d, we train a classifier $g^{(d)}$ exactly as described in §3.3, and define a predictor of $Y_k^{(d)}$ by

$$f_k^{(d)}(I)=g^{(d)}\left(\mathbf{X}^{(d)}(k,I)\right).$$

The hierarchy is "unfolded" for testing and the predictors are evaluated in an adaptive way by visiting the nodes (cells) according to breadth-first "coarse-to-fine" search. A classifier is evaluated if and only if all its ancestors along the branch up to its root have been evaluated and returned a positive response. In particular, once a classifier at a node responds negatively, none of the descendant classifiers are ever evaluated. The result of the detection process is the list of leaves which are reached and respond positively. In this way, pose cells corresponding to obvious non-target regions such as flat areas are discarded early in the search and the computation is invested the ambiguous areas, for example, parts of images with "cat-like" shape or texture.

5. Base Classifier

As described in §4, given a family of pose-indexed features and a hierarchical partitioning of the pose space, we build a binary classifier $g^{(d)}$ for each level d in the hierarchy, trained from a set of examples of the type described in §3.3. In this section we describe that classifier, dropping the superscript d for clarity. The actual parameter values we used for the experiments on cat detection are given in §6.5.

Evidently, inducing such a mapping g is a standard machine learning problem. A simple candidate is a thresholded linear combination of V stumps trained with Adaboost (Freund and Schapire, 1999):

$$g(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{V} \alpha_i 1_{\{x^{\delta_i} \geq \tau_i\}} \geq \rho \\ 0 & \text{otherwise.} \end{cases}$$

Here, x^j is the j'th coordinate of the feature vector.

For any given true positive rate η, the threshold ρ in g, and more generally the thresholds $\rho^{(d)}$ in $g^{(d)}, 1 \leq d \leq D$, are chosen to achieve on a validation set a targeted decreasing sequence of true positive rates yielding η.

To select the stumps, that is the α_i, δ_i and τ_i, special attention must be given to the highly unbalanced populations we are dealing with. Of course in our detection problem, the prior distribution is very skewed, with an extremely low probability of the presence of a target at a pose picked at random. Correspondingly, the number of samples we have from the positive population (cats in our case) is orders of magnitude smaller than the number of samples we can easily assemble from the negative population. Still, any tractable sampling of the negative population is still too sparse to account for the negative sub-population which lives close to positive examples. To address these issues, we propose a variation the standard weighting-by-sampling in order to approximate standard Adaboost using a training set containing one million negative examples.

The popular cascade approach handles that dilemma with bootstrapping: training each level with a sample of negative examples which survive the filtering of the previous classifiers in the cascade. In this way the sampling is eventually concentrated on the "difficult" negative samples. This is similar in practice to what boosting itself is intended to do, namely ignore easily classified samples and concentrate on the difficult ones. We avoid the complexity of tuning such a cascade by using all the negative examples through an asymmetric, sampling-based version of standard boosting. This provides an excellent approximation to the exact weighting for a fraction of the computational cost.

When picking a stump we approximate the weighted error with an error computed over all positive samples and a *random subset of negative samples* drawn according to the current boosting weights. Hence, we keep the response of the strong classifier up-to-date on $S \simeq 10^6$ samples, but we pick the optimal weak learners at every step based on $M \simeq 10^4$ samples.

More precisely, at a certain iteration of the boosting procedure, let ω_s denote the weight of sample $s = 1, \ldots, S$, let $Y_s \in \{0,1\}$ be its true class, and let

$$\omega_{neg} = \sum_s \omega_s \mathbf{1}_{\{Y_s = 0\}}$$

be the total weight of the negative samples. We sample independently M indices $S_1,\ldots,S_M$ in $\{1,\ldots,S\}$ according to the negative sample density

$$P(S_m = s) = \frac{\omega_s \mathbf{1}_{\{Y_s=0\}}}{\omega_{neg}},\ m = 1,\ldots,M.$$

Then we re-weight the training samples as follows:

$$\omega'_s = \begin{cases} \omega_s & \text{if } Y_s = 1 \\ \omega_{neg}\dfrac{\|\{m : S_m = s\}\|}{M} & \text{otherwise.} \end{cases}$$

This can be seen as an approximation to the distribution on the full training set obtained by (1) keeping all positive samples with their original weights, and (2) selecting a random subset of negative samples according to their original weights, and giving them a uniform weight.

However, sampling the negative examples at every boosting step is computationally very expensive, mainly because it requires loading into memory all the training images, and extracting the edge features at multiple scales. Consequently, we decompose the total number of stumps V into B blocks of U stumps, and run this sampling strategy at the beginning of every block. We also sample a subset of R features among the millions of features available at the beginning of every block. Hence, our overall learning process can be seen as a standard boosting procedure decomposed into B blocks of U steps. At the beginning of every block, we sample M negative samples among S according to their current boosting weight and we sample R features uniformly among the Q to consider. We then run U boosting steps based on these features and these training samples.

This process ensures that any sample for which the classifier response is strongly incorrect will eventually be picked. In our experiments we sample ten negative examples for every positive example. From a computational perspective this sampling is negligible as it only accounts for about 1% of the total training time.

6. Cat Detection

We now specialize everything above to cat detection. The original training images and available ground truth are described in §6.1. Then, in §6.2, we define a family of highly robust, base image features based on counting edge frequencies over rectangular areas, and in §6.3 we propose a way to index such features with the pose of a cat defined by its head location and scale and belly location. The specific pose cell hierarchy is described in §6.4 and choice of parameters for the classifiers in §6.5. Finally, the results of our experiments are presented in §6.6, including the similarity criteria used for performance evaluation, the post-processing applied to the raw detections in order to reduce "duplicate" detections and the manner of computing ROC curves.

6.1 Cat Images and Poses

The cat images were randomly sampled from the web site RateMyKitten[1] and can be downloaded from `http://www.idiap.ch/folded-ctf`. Images of cluttered scenes without cats, mostly home interiors, were sampled from various web sites. The complete database we are using has 2,330 images containing a total of 1,988 cats.

1. Web site can be found at `http://www.ratemykitten.com`.

Figure 6: Each target is labeled by hand with a pose $(h,r,b) \in \mathcal{Z} \times \mathbb{R}^+ \times \mathcal{Z}$ specifying the head location, the head radius and the belly location.

Each cat was manually annotated with one circle roughly outlining the head, from which the head size (diameter) and head location (center) are derived, and one point placed more or less, quite subjectively, at the center of mass, which we have referred to as the "belly" location (see Figure 6). Hence, the pose of a cat is (h,r,b) where h is the location in the image plane $\mathcal{Z}$ of the center of the head, r its radius, and b is the belly location.

For each experiment, we split this database at random into a training set containing 75% of the images and a test set containing the other 25%. Two-thirds of the training set are used for choosing the weak learners and one-third for the thresholds.

6.2 Base Image Features

As described in §3, the pose-indexed features are defined in terms of base image features. First, an image is processed by computing, at every location $z \in \mathcal{Z}$, the responses of eight edge-detectors similar to those proposed in Amit et al. (1998) (see Figures 7 and 8), but at three different scales, ending up with 24 Boolean features $e_1(z), \ldots, e_{24}(z)$ corresponding to four orientations and two polarities. In addition, we add a variance-based binary test $e_0(z)$ which responds positively if the variance of the gray levels in a 16×16 neighborhood of z exceeds a fixed threshold. Our features are based on counting the number of responses of these 25 detectors over a rectangular areas (Fan, 2006), which can be done in constant time by using 25 integral images (Simard et al., 1999).

From these edge maps and the raw gray levels we define the following three types of base image features:

1. **Edge proportion:** The proportion of an edge type in a rectangular window. Given a rectangular window W and an edge type $\lambda \in \{0,1,\ldots,24\}$, the response is the number of pixels z in W for which $e_\lambda(z) = 1$, divided by the total number of pixels in W if $\lambda = 0$ or by the number of pixels in W for which $e_0(z) = 1$ if $\lambda > 0$.

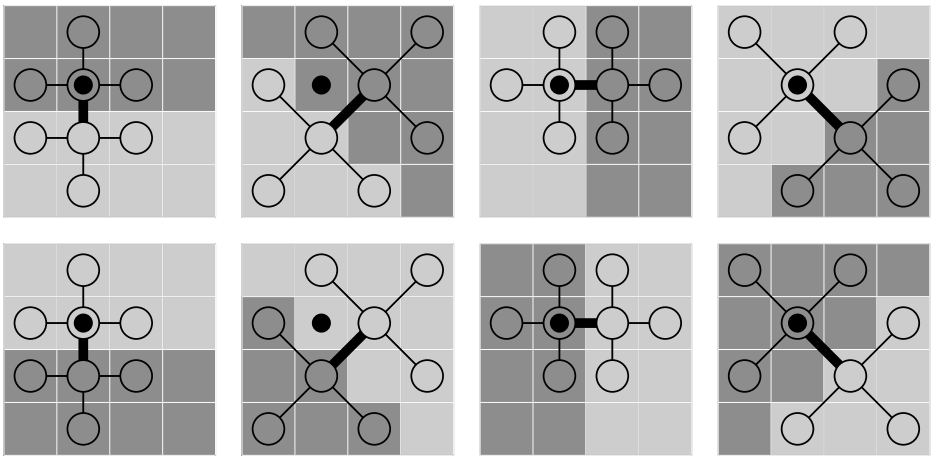


Figure 7: Our edge-detectors: For each of four orientations and two polarities, an edge is detected at a certain location (the dark circle) if the absolute difference between the intensities of the two pixels linked by the thick segment is greater than each of the six intensity differences for pixels connected by a thin segment.

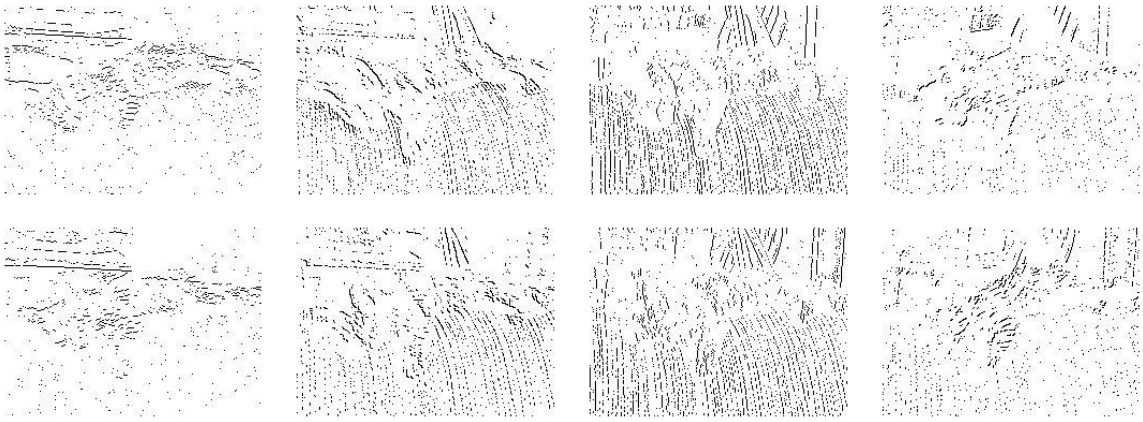


Figure 8: Result of the edge detector. Each one of the eight binary images in the two bottom rows corresponds to one orientation of the edge detectors of Figure 7.

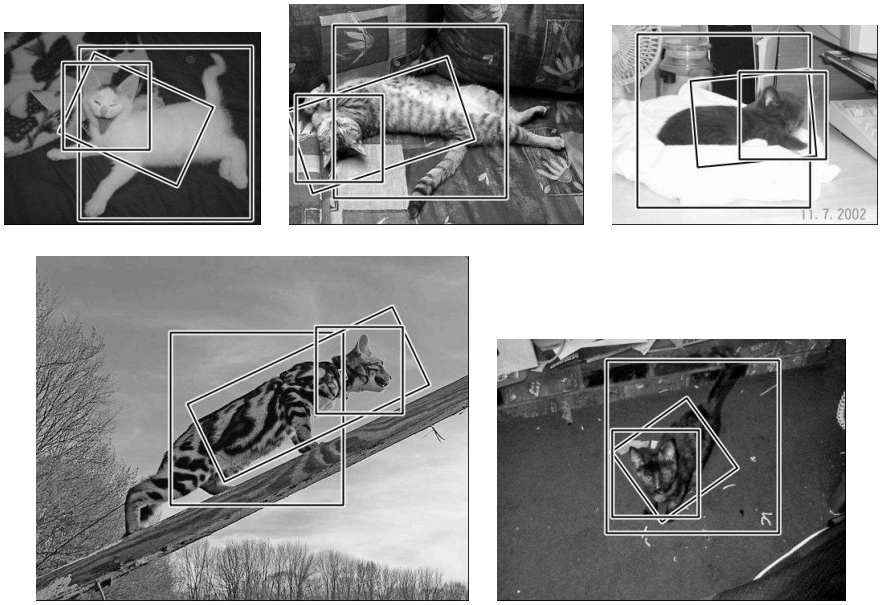


Figure 9: From the centroid of any pose cell, we define three reference frames: The *head frame* is centered on the head center, of size twice the head size; the *belly frame* is centered on the belly and of size four times the head size; the *head-belly frame* is centered on the middle point between the head and the belly, of height twice the head size, of width twice the distance between the head and the belly, and is tilted accordingly.

2. **Edge orientation histogram distance:** Given again two rectangular windows W_1 and W_2, and a scale s, the response is the L^1 norm between the empirical eight-bin histograms of orientations corresponding to the eight edge detectors at scale s.

3. **Gray-scale histogram distance:** Given two rectangular windows W_1 and W_2, the response is the L^1 norm between the sixteen-bin empirical histograms of gray-scales for the two windows.

The rational behind the features of type 1 is to endow the classifiers with the ability to check for the presence of certain pieces of outlines or textures. The motivation for types 2 and 3 is to offer the capability of checking for similarity in either edge or gray-scale statistics between different parts of the image, typically to check for a silhouette in the case of very blurry contours. Some examples of features actually picked during the training are shown in Figures 10 and 11.

6.3 Indexing Features by Pose

As formalized in §3.3, a pose-indexed feature is a real-valued function of both a pose cell and an image. The features described in the previous section are standard functionals of the image alone. Since the response of any of them depends on counting certain edge types over rectangular windows in the image, we construct our family of pose-indexed features indirectly by indexing both the edge types and the window locations with the pose cell.

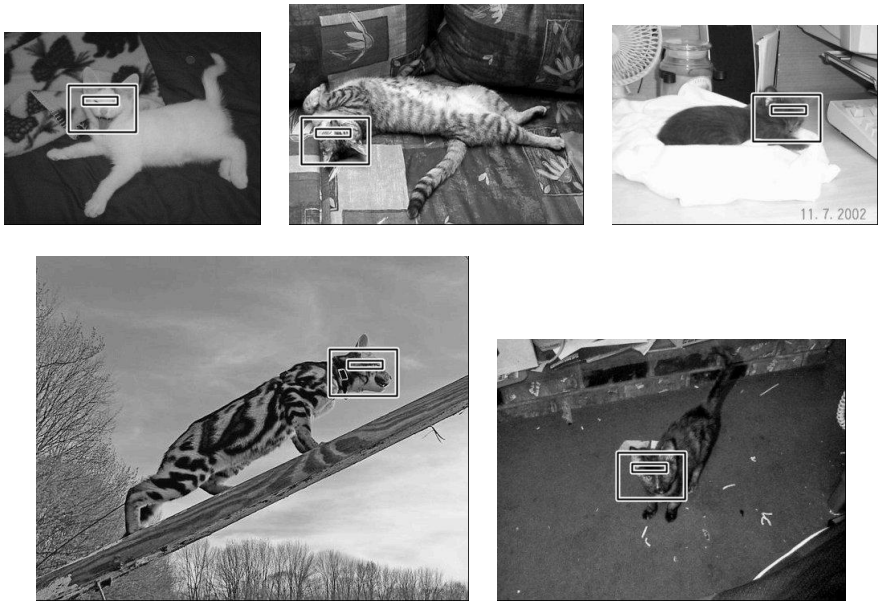


Figure 10: Registration on the true poses of the first feature selected in the HB detector (see §6.5), which compares edge orientation histograms. Both windows are defined relative to the head frame.

For any pose cell index k, we compute the average head location $h = h_k$, the average belly location $b = b_k$, and the average head radius $r = r_k$ of the pose cell $\mathcal{Y}_k$. From these parameters we compute three reference frames, as shown on Figure 9:

1. The **head frame** is a square centered on h and of size $4r$.
2. The **belly frame** is a square centered on b and of size $8r$.
3. The **head-belly frame** is a rectangle centered on the midpoint of h and b, tilted accordingly, and of height $4r$ and width twice $\|h-b\|$.

Note that the definition of such a frame actually involves the definition of a vector basis, hence an orientation. The three types of frames are oriented according to the relative horizontal locations of the head and belly of the cat, so a reflection around a horizontal axis of the image, hence of the cat pose, would move the points defined in these frames consistently.

We add to the parameterization of each feature window a discrete parameter specifying in which of these three reference frames the window is defined. Windows relative to the head or the belly frame are simply translated and scaled accordingly. Windows relative to the head-belly frame are translated so that their centers remain fixed in the frame, and are scaled according to the average of the height and width of the frame. See Figures 10 and 11.

Finally, we add another binary flag to windows defined in the head-belly frame to specify if the edge detectors are also registered. In that case, the orientation of the edges is rotated according to the tilt of the head-belly frame.

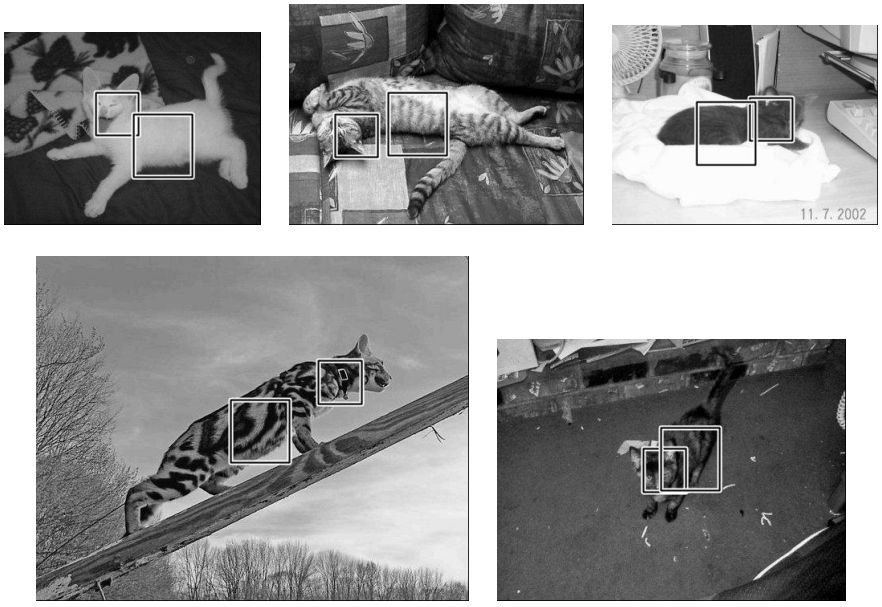


Figure 11: Registration on the true poses of the third feature selected in the second level of the HB detector (see §6.5), which compares grayscale histograms. One window is relative to the head-belly frame, and the second one to the belly frame.

6.4 Hierarchy of Poses

We only consider triples (h,r,b) which are consistent with the relative locations seen on the training set. For instance, this discards poses with very large ratios $\|h-b\|/r$. However, h and b may be very close together, for example when the belly is behind the head, or very far apart, for example when the cat is stretched out. Hence the full pose space is $\mathcal{Y} \subset \mathcal{Z} \times \mathbb{R}^+ \times \mathcal{Z}$.

We use a hierarchy with only $D = 2$ levels in order to concentrate on folded learning with stationary features. The first level $\{\mathcal{Y}_1^{(1)}, \ldots, \mathcal{Y}_{K_1}^{(1)}\}$ is based on first restricting the head radius to $[25, 200]$, and on splitting that domain into 15 sub-intervals of the form $[r, 2^{1/5} r]$. For each such scale interval, we divide the full lattice $\mathcal{Z}$ into non-overlapping regular squares of the form $[x_h, x_h + r/5] \times [y_h, y_h + r/5]$. This procedure creates $K_1 \simeq 50,000$ head parameter cells $[x_h, x_h + r/5] \times [y_h, y_h + r/5] \times [r, 2^{1/5} r]$ for a 640×480 image. For any such cell, the admissible domain for the belly locations is the convex envelope of the belly locations seen in the training examples, normalized in location and scale with respect to the head location and radius.

The second level $\{\mathcal{Y}_1^{(2)}, \ldots, \mathcal{Y}_{K_2}^{(2)}\}$ is obtained by splitting the belly location domain into regular squares $[x_b, x_b + r/2] \times [y_b, y_b + r/2]$. There are $\simeq 500$ such belly squares, hence the total number of pose cells in the second level is $K_2 \simeq 2.5 \times 10^7$.

The top-left illustration in Figure 12 depicts the cells in the first level of the hierarchy as open circles and cells in the second level as black dot connected to an open circle "kept alive" during processing the first level. More specifically, as shown in Figure 12, the algorithmic process corresponding to this two-level hierarchy is as follows:

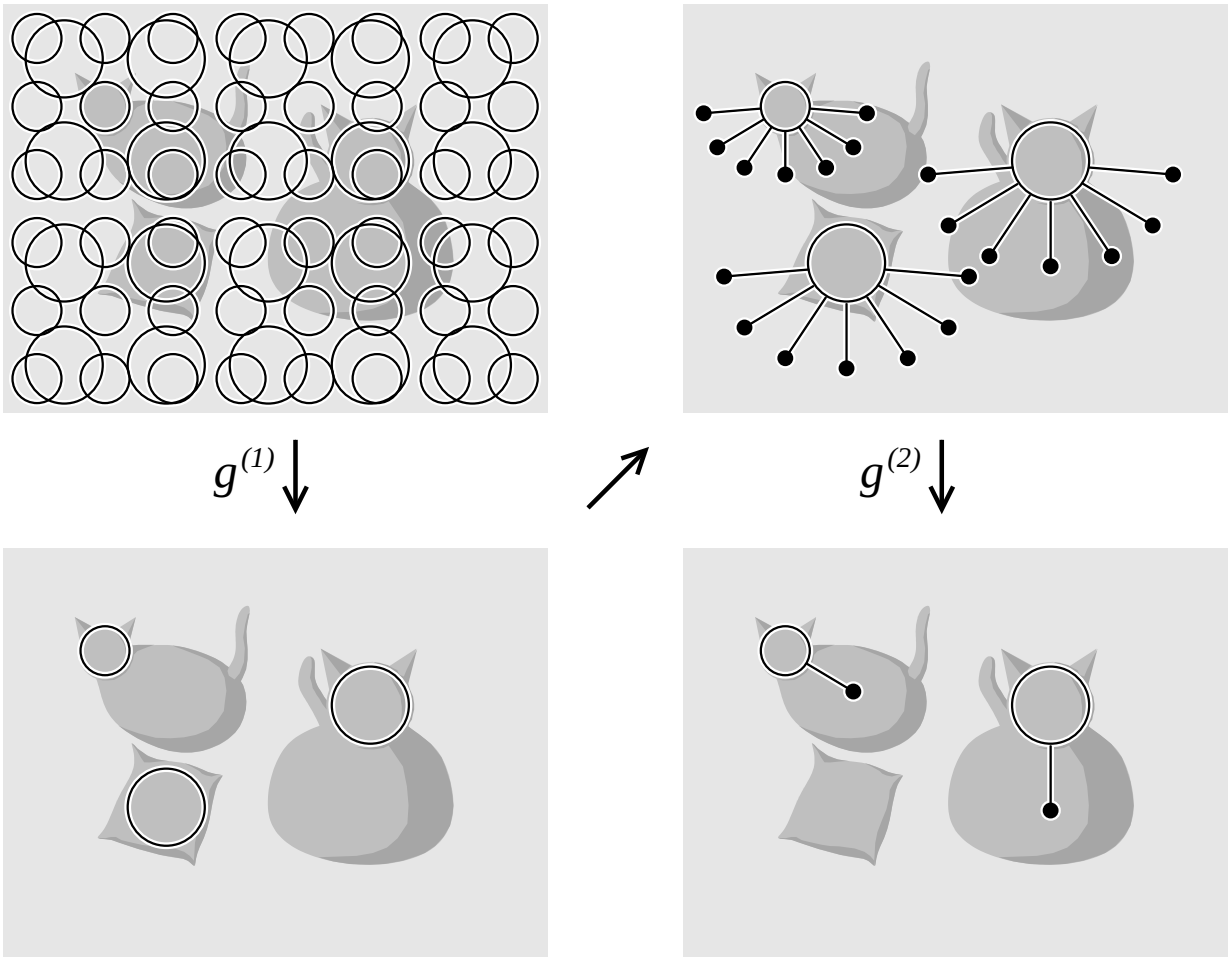


Figure 12: Parsing a scene with a two-level hierarchy to find cats: First, a classifier $g^{(1)}$ is evaluated over a sublattice of possible head locations and all alarms above a very low threshold are retained. Then a classifier $g^{(2)}$ is evaluated for each pair of head-belly locations on a sublattice consistent with the retained head alarms and with observed statistics about joint head-belly locations. For clarity, the depicted discretization of the pose space is idealized, and far coarser than in the actual experiments; for an image of size 640×480 pixels, we consider $\simeq 50,000$ head pose cells and $\simeq 2.5 \times 10^7$ head-belly pose cells.

1. The first stage loops over a sublattice of possible head locations and scales in the scene, evaluates the response of the appropriate first-level classifier and retains all alarms using a very low (i.e., conservative) threshold.

2. The second stage visits each location and scale tagged by the first stage, scans a sublattice of all "consistent" belly locations (all those actually observed on training images) and evaluates an appropriate second-level classifier for every such candidate pair of locations.

6.5 Detectors

Whereas our aim is to detect both the head and the body, detecting the head alone is similar to the well-studied problem of detecting frontal views of human faces. As stated earlier, if the pose reduces to a single position, data-aggregation is straightforward by translating either whole images or features. Still, detecting cat heads is a logical first step in trying to find cats since the head is clearly the most stable landmark and the part of the cat with the least variation, assuming of course that the head is visible, which is the case with our data (for the same reason that family photographs display the faces of people). Moreover, comparing the performance of varying strategies (field of view, "checking" for the belly separately, demanding "consistency", etc.) provides some insight on the nature of the problem and serves as a simple way of demonstrating the power of the base feature set and the asymmetrical weighting by sampling. Detecting heads alone does not, however, expose the full strength of the folded hierarchy; for that we need to address the harder task of accurately estimating (h,r,b) for the visible cats, our core objective, and for which we will compare our pose-indexed method with a more standard parts-based detector.

In all the experiments we present, the classifiers are trained as described in §5, with $B = 25$ blocks of $U = 100$ stumps (thresholded features), and we optimize over a sample of $R = 10,000$ pose-indexed features in every such block. The total number of negative samples we consider is $S \simeq 10^6$, and we sample $M \simeq 10^4$ of these per block.

In measuring performance, we consider the two following detections strategies:

- **H+B** is a standard parts detector, implemented adaptively. The "+" between H and B indicates that the two part detectors are trained separately.

  The first level classifier $g^{(1)}$ can only use pose-indexed feature defined relatively to the head frame and the second level classifier $g^{(2)}$ can only use pose-indexed features defined relatively to the belly frame. Since that second-level detector is designed not to exploit the information in the joint locations of the head and belly, the frames here have fixed orientation, and reflecting the cat pose horizontally would move but not invert the frames. See §6.3 for details about the orientations of the frames.

- **HB** is the hierarchical detector based on the two-level hierarchy and folded learning.

  The difference with H+B is that HB uses stationary features in the second level which can be defined relatively to any of the three reference frames (head, belly or head-belly) in order to take into account the position of the head in searching for the belly. For instance, a pose-indexed features in this detector could compare the texture between a patch located on the head and a patch located on the belly.

6.6 Results

In order to be precise about what a constitutes a true detection, we define two criteria of similarity. We say that two poses (h,r,b) and (h',r',b') **collide** if (1) The head radii are very similar: $1/1.25 \leq r/r' \leq 1.25$; and (2) Either the head or belly locations are close: $\min(\|h-h'\|, \|b-b'\|) \leq 0.25\sqrt{rr'}$. And we will say that two poses are **similar** if (1) The head radii are similar: $1/1.5 \leq r/r' \leq 1.5$; (2) the head locations are nearby each other: $\|h-h'\| \leq 2\sqrt{rr'}$; and (3) the belly locations are nearby each other: $\|b-b'\| \leq 4\sqrt{rr'}$. See Figure 13.

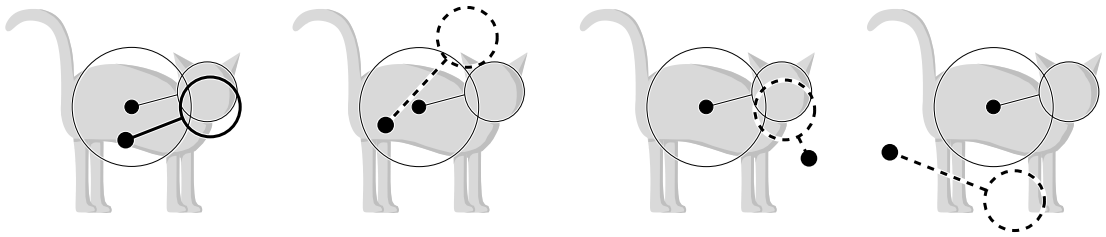


Figure 13: Two alarms are considered as similar if the head radii are similar and if, as shown on this figure, the distance between the two head locations is less than the average head radius, and if the distance between the belly locations is less than twice the average head radius. See §6.6. Based on that criterion, if the true pose is the one shown in thin lines and the thick poses are detections, only the leftmost one would be counted as a true hit. The three others, shown in dashed lines, would be counted as false alarms.

Given these two criteria, the alarms kept after thresholding the classifier responses are post-processed with a crude clustering. We visit the alarms in the order of the response of the detector, and for each alarm we remove all others that collide with it. Then we visit these surviving alarms again in the order of the response and for each alarm we remove all the other alarms which are similar.

The procedure we use to produce ROC curves is the following. We run ten rounds in which the training and test images are selected at random, and in each round we estimate the classifier thresholds for achieving ten different true-positive rates η (see § 5). Hence, we generate 100 pairs of rates, each consisting of a true-positive rate and an average number of false alarms per image. An alarm is counted as true positive if there exists a cat in the image with a similar pose according to the criterion described above.

The error rates in Figure 2 and Table 2 demonstrate the power of conditioning on the full pose. Using stationary features to build classifiers dedicated to fine cells allows the search for one part to be informed by the location of the other, and allows for consistency checks. This is more discriminating than checking for individual parts separately. Indeed, the error rates are cut be a factor of roughly two at very high true-positive rates and a factor of three at lower true-positive rates. It should be emphasized as well that even the weaker ROC curve is impressive in absolute terms, which affirms the efficacy of even the naive stationary features used by the H+B detector and the modified boosting strategy for learning.

An example of how features selected in the second-level of the HB classifier exploit the full pose can be seen in Figure 11. Such a feature allows the HB detector to check for highly discriminating properties of the data, such as the continuity of appearance between the head and the belly, or discontinuities in the direction orthogonal to the head-belly axis.

More then two-thirds of the false positives are located on or very near cats; see Figure 14. Such false positives are exceedingly difficult to filter out. For instance, a false head detection lying around or on the belly will be supported by the second-level classifier because the location of the true belly will usually be visited.

TP	H+B	HB
90%	12.84	5.85
80%	3.53	1.63
70%	1.35	0.50
60%	0.61	0.23
50%	0.33	0.12
40%	0.18	0.06
30%	0.10	0.03

Table 2: Average number of false alarms per images of size 640×480 vs. the true positive rate for the head-belly detection, as defined by the similarity criterion of §6.6 and Figure 13.

Finally, we performed a similar experiment by testing the classifiers trained on the Ratemykitten data set on a sample of cat images chosen from the PASCAL VOC2007 challenge set images.[2] The PASCAL data set was assembled for evaluating methods for *classification*, that is, labeling an entire image according to one of the object categories, rather than methods for object detection and localization. There are 332 cat images in the PASCAL set; our test set consists of those 201 images for which the body is at least partially visible. This provides an even more challenging test set than the images from Ratemykitten and the performance of our classifier is somewhat reduced. For instance, at a true positive rate of 51%, the average number of false alarms per image of size 640×480 is 0.9. The results on a random sample of twenty of the 201 test images is shown in Figure 15.

7. Conclusion

We have presented a novel detection algorithm for objects with a complex pose. Our main contribution is the idea of stationary, pose-indexed features, a variation on deformable templates without whole image transforms. This makes it possible to train pose-specific classifiers without clustering the data, and hence without reducing the number of training examples. Moreover, combining simultaneous training with a sequential exploration of the pose space overcomes the main drawback of previous coarse-to-fine strategies, especially for going beyond scale and translation. Unlike in earlier variations, graded, tree-structured representations can now be learned efficiently because there is only one classifier to train per level of the hierarchy rather than one per node.

We have illustrated these stationary features by detecting cats in cluttered still images. As indicated earlier, the data are available at `http://www.idiap.ch/folded-ctf`. We chose boosting with edge and intensity counts, but any base learning algorithm and any flexible base feature set could be used. Indeed, the framework can accommodate very general features, for instance the average color or average response of any local feature in an area defined by the pose. The resulting algorithm is a two-stage process, first visiting potential head locations alone and then examining additional aspects of the pose consistent with and informed by candidate head locations.

In principle, our approach can deal with very complex detection problems in which multiple objects of interest are parametrized by a rich hidden pose. However, two basic limitations must

2. Website can be found at `http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/`.

Figure 14: Detection results with stationary features and a folded two-level hierarchy on scenes picked uniformly at random in the RateMykitten test set, with a true-positive rate of 71%. The circle shows the estimated head size and location, and the dot the estimated belly location.

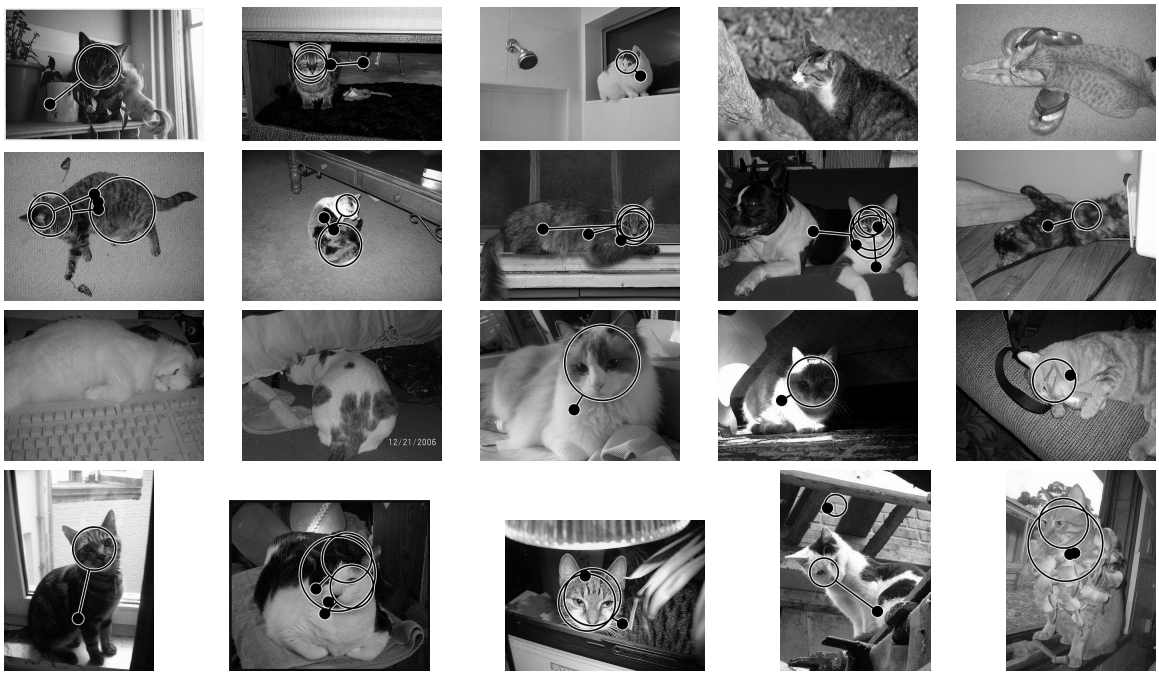


Figure 15: Detection results with stationary features and a folded two-level hierarchy on scenes picked uniformly at random in the PASCAL VOC2007 challenge test set, with a true-positive rate of 50%. The circle shows the estimated head size and location, and the dot the estimated belly location.

first be addressed. The first is the design of adequate stationary features. Whereas difficult, this is far simpler than the search for full geometric invariants. Since the hidden state is explicitly examined in traversing the hierarchy, there is no need to integrate over all possible values of the hidden quantities. The second difficulty is labeling a training set with rich ground truth. One way to tackle this problem is by exploiting other information available during training, for instance temporal consistency if there are motion data. Our viewpoint is that small, richly annotated, training sets are at least as appealing for general learning as large ones with minimal annotation.

Acknowledgments

The work of FF was partially supported by the Swiss National Science Foundation under the National Centre of Competence in Research (NCCR) on "Interactive Multimodal Information Management" (IM2). The work of DG was partially supported by the National Science Foundation under grants NSF 0427223 and NSF 0625687, and by the Office of Naval Research under contract N00014-07-1-1002.

We are also extremely grateful to the reviewers for their thoughtful suggestions, as well as to the web site `http://www.ratemykitten.com`, and to Harrison Page in particular, for providing us with the remarkable set of cat images.

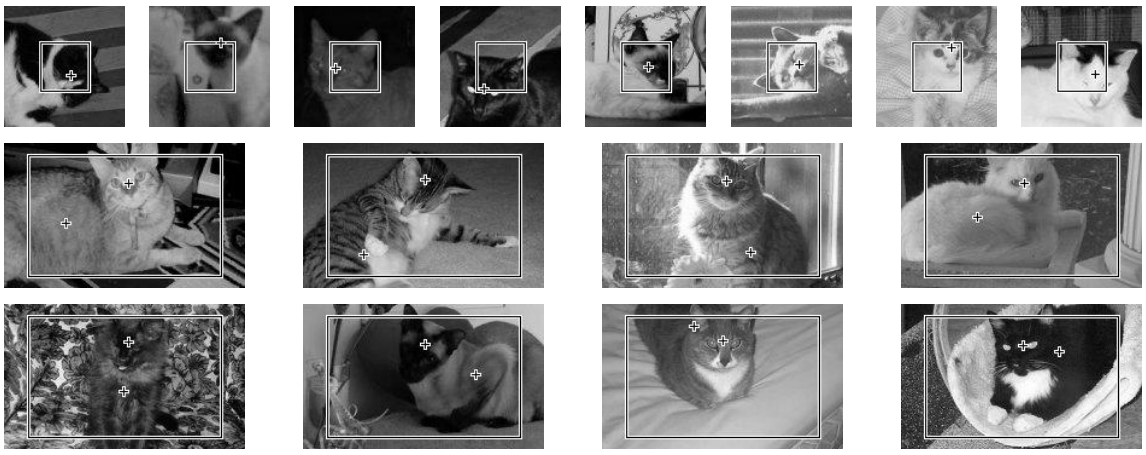


Figure 16: A few positive examples picked uniformly at random in the simplified setting of the motivational experiment. Top row: samples from the head experiments. Bottom two rows: samples from the head-belly experiments. The crosses depict the head and belly centers provided on the training data. The boxes show the admissible pose domain $\mathcal{Y}$.

Appendix A. Quantifying Trade-offs

We summarize two series of experiments designed to study the impact on accuracy of data-fragmentation, with and without controlling for total online computation. In both series the goal is to predict the presence of a target with high accuracy in pose. A more detailed account of these experiments can be found in Fleuret and Geman (2007).

A.1 Settings

Since training with fragmentation is not feasible for any complete partition of a complex pose space at a realistic resolution, the images we consider in these experiments have been cropped from the original data set so that the pose space $\mathcal{Y}$ is already strongly constrained.

In the first series of experiments the target pose is the center of the cat head, constrained to $\mathcal{Y} = [-20,20] \times [-20,20]$ in a 100×100 image. It is this pose space that will be investigated at different resolutions. The top row of Figure 16 shows a few of these scenes with a target. In the second series of experiments the pose is the *pair of locations* (h,b) for the head and belly, constrained to $\mathcal{Y} = ([0,5] \times [0,5]) \times ([-80,80] \times [-20,80])$ in a 200×140 image centered at the square. The two bottom rows of Figure 16 show a few of these scenes with a target.

In both series, our objective is to compare the performance of classifiers when the training data are either fragmented or aggregated and when the computational cost is either equalized or not. More precisely, we consider three partitions of $\mathcal{Y}$ into $K = 1$, 4 and 16 pose cells. In each series, we build four detection systems. Three of them are trained under data-fragmentation at the three considered resolutions, namely $K = 1$, $K = 4$ or $K = 16$ pose cells. The fourth classifier is trained with the pose-indexed, stationary features at the finest resolution $K = 16$. The stationary features are based on the head frame alone for the head experiments, and on both the head frame and the head-belly frame for the head-belly experiments.

The computational cost for evaluating one such classifier is proportional to the number of stumps it combines. In the particular case of boosting, a classifier combining only a fixed number of weak learners is still effective, and hence, unlike many discriminative methods, computation is easy to control. This motivates a very simple strategy to equalize the cost among experiments: We simply control the total number of feature evaluations.

As a measure of performance, we estimate the number of false alarms for any given true positive rate. In order to compare results across resolutions, the labeling of detections as true or false positives occurs at the coarsest resolution. For simplicity, for the head-belly case, we only score the estimated head location.

A.2 Results

The results demonstrate the gain in performance in constraining the population provided there is no fragmentation of the data. In the head experiments, even with fragmentation, higher resolution results in fewer false alarms. The improvement is marginal at high true positive rates, but increases to two-fold for a true positive rate of 70%. This is not true for the head-belly experiments, where sixteen pose cells do worse than four, with or without cost equalization, which can be explained to some extent by the lower variation in the appearance of cat heads than full cat bodies, and hence fewer samples may be sufficient for accurate head detection.

As expected, without controlling the on-line computational cost, aggregation with stationary features is more discriminating than the fragmented classifiers in both experiments and at any true positive rate, reducing the false positive rate by a factor of three to five. Still, the performance of the classifiers when cost is equalized shows the influence of computation in this framework: at the finest resolution, the number of false alarms in the head experiments increases by a factor greater than four at any true-positive rate, and by two orders of magnitude in the head-belly experiments.

These results also demonstrate the pivotal role of computation if we are to extend this approach to a realistically fine partition of a complex pose space. Consider an image of resolution 640×480 and a single scale range for the head. Obtaining an accuracy in the locations of the head and the belly of five pixels requires more than 7×10^6 pose cells. Investing computation *uniformly* among cells is therefore hopeless, and argues for an adaptive strategy able to distribute computation in a highly special and uneven manner.

The conclusions drawn can be summarized in two key points:

1. **The need for data-aggregation**: *Dealing with a rich pose by training specialized predictors from constrained sub-populations is not feasible, both in terms of offline computation and sample size requirements. Aggregation of data using stationary features appears to be a sound strategy to overcome the sample size dilemma as it transfers the burden of learning to the design of the features.*

2. **The need for adaptive search**: *If fragmentation can be avoided and a single classifier built from all the data and analytically transformed into dedicated classifiers, the computation necessary to cover a partition of a pose space of reasonable accuracy is not realistic if the effort is uniformly distributed over cells.*

As indicated, stationary features provide a coherent strategy for dealing with data-aggregation but do not resolve the computational dilemma resulting from investigating many possible poses during scene processing. Hierarchical representations largely do.

References

Y. Amit, D. Geman, and B. Jedynak. Efficient focusing and face detection. In *Face Recognition: From Theory to Applications*. Springer Verlag, 1998.

D. J. Crandall and D. P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *European Conference on Computer Vision*, pages 16–29, 2006.

X. Fan. *Learning a Hierarchy of Classifiers for Multi-class Shape Detection*. PhD thesis, Johns Hopkins University, 2006.

F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal of Computer Vision (IJCV)*, 41(1/2):85–107, 2001.

F. Fleuret and D. Geman. Stationary features and cat detection. Technical Report 07-56, IDIAP Research Institute, October 2007.

Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):771–780, 1999.

S. Gangaputra and D. Geman. A design principle for coarse-to-fine classification. In *Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1877–1884, 2006.

D.M. Gavrila. Multi-frame hierarchical template matching using distance transforms. In *International Conference on Pattern Recognition*, 1998.

S. Geman, K. Manbeck, and E. McClure. Coarse-to-fine search and rank-sum statistics in object recognition. Technical report, Brown University, 1995.

S. Geman, D. F. Potter, and Z. Chi. Composition systems. *Quarterly of Applied Mathematics*, LX: 707–736, 2002.

U. Grenander. *General Pattern Theory*. Oxford U. Press, 1993.

D. Huttenlocher and P. Felzenszwalb. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

D.P. Huttenlocher and W.J. Rucklidge. A multi-resolution technique for comparing images using the hausdorff distance. In *Conference on Computer Vision and Pattern Recognition*, 1993.

Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Conference on Computer Vision and Pattern Recognition*. IEEE Press, 2004.

F. Li, R. Fergus, and P. Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *International Conference on Computer Vision*, volume 2, page 1134, 2003.

J.L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

B. Ommer, M. Sauter, and J. M. Buhmann. Learning top-down grouping of compositional hierarchies for recognition. In *Conference on Computer Vision and Pattern Recognition*, 2006.

C. Papageorgiou and T. Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, June 2000.

H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–28, 1998.

H. Schneiderman and T. Kanade. Object detection using the statistics of parts. *International Journal of Computer Vision*, 56(3):151–177, 2004.

P. Simard, L. Bottou, P. Haffner, and Y. LeCun. Boxlets: a fast convolution algorithm for neural networks and signal processing. In *Neural Information Processing Systems*, volume 11, 1999.

B. Stenger, A. Thayananthan, P.H.S. Torr, and R. Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28 (9):1372–1384, 2006.

P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:369–382, 2008.

S.C. Zhu and D. Mumford. *A Stochastic Grammar of Images*, volume 2 of *Foundations and Trends in Computer Graphics and Vision*, pages 259–362. Now Publishers, 2006.

Journal of Machine Learning Research 9 (2008) 2579-2605 Submitted 5/08; Revised 9/08; Published 11/08

Visualizing Data using t-SNE

Laurens van der Maaten LVDMAATEN@GMAIL.COM
TiCC
Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

Editor: Yoshua Bengio

Abstract

We present a new technique called "t-SNE" that visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding (Hinton and Roweis, 2002) that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map. t-SNE is better than existing techniques at creating a single map that reveals structure at many different scales. This is particularly important for high-dimensional data that lie on several different, but related, low-dimensional manifolds, such as images of objects from multiple classes seen from multiple viewpoints. For visualizing the structure of very large data sets, we show how t-SNE can use random walks on neighborhood graphs to allow the implicit structure of all of the data to influence the way in which a subset of the data is displayed. We illustrate the performance of t-SNE on a wide variety of data sets and compare it with many other non-parametric visualization techniques, including Sammon mapping, Isomap, and Locally Linear Embedding. The visualizations produced by t-SNE are significantly better than those produced by the other techniques on almost all of the data sets.

Keywords: visualization, dimensionality reduction, manifold learning, embedding algorithms, multidimensional scaling

1. Introduction

Visualization of high-dimensional data is an important problem in many different domains, and deals with data of widely varying dimensionality. Cell nuclei that are relevant to breast cancer, for example, are described by approximately 30 variables (Street et al., 1993), whereas the pixel intensity vectors used to represent images or the word-count vectors used to represent documents typically have thousands of dimensions. Over the last few decades, a variety of techniques for the visualization of such high-dimensional data have been proposed, many of which are reviewed by de Oliveira and Levkowitz (2003). Important techniques include iconographic displays such as Chernoff faces (Chernoff, 1973), pixel-based techniques (Keim, 2000), and techniques that represent the dimensions in the data as vertices in a graph (Battista et al., 1994). Most of these techniques simply provide tools to display more than two data dimensions, and leave the interpretation of the

data to the human observer. This severely limits the applicability of these techniques to real-world data sets that contain thousands of high-dimensional datapoints.

In contrast to the visualization techniques discussed above, dimensionality reduction methods convert the high-dimensional data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ into two or three-dimensional data $\mathcal{Y} = \{y_1, y_2, ..., y_n\}$ that can be displayed in a scatterplot. In the paper, we refer to the low-dimensional data representation $\mathcal{Y}$ as a map, and to the low-dimensional representations y_i of individual datapoints as map points. The aim of dimensionality reduction is to preserve as much of the significant structure of the high-dimensional data as possible in the low-dimensional map. Various techniques for this problem have been proposed that differ in the type of structure they preserve. Traditional dimensionality reduction techniques such as Principal Components Analysis (PCA; Hotelling, 1933) and classical multidimensional scaling (MDS; Torgerson, 1952) are linear techniques that focus on keeping the low-dimensional representations of dissimilar datapoints far apart. For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to keep the low-dimensional representations of very similar datapoints close together, which is typically not possible with a linear mapping.

A large number of nonlinear dimensionality reduction techniques that aim to preserve the local structure of data have been proposed, many of which are reviewed by Lee and Verleysen (2007). In particular, we mention the following seven techniques: (1) Sammon mapping (Sammon, 1969), (2) curvilinear components analysis (CCA; Demartines and Hérault, 1997), (3) Stochastic Neighbor Embedding (SNE; Hinton and Roweis, 2002), (4) Isomap (Tenenbaum et al., 2000), (5) Maximum Variance Unfolding (MVU; Weinberger et al., 2004), (6) Locally Linear Embedding (LLE; Roweis and Saul, 2000), and (7) Laplacian Eigenmaps (Belkin and Niyogi, 2002). Despite the strong performance of these techniques on artificial data sets, they are often not very successful at visualizing real, high-dimensional data. In particular, most of the techniques are not capable of retaining both the local and the global structure of the data in a single map. For instance, a recent study reveals that even a semi-supervised variant of MVU is not capable of separating handwritten digits into their natural clusters (Song et al., 2007).

In this paper, we describe a way of converting a high-dimensional data set into a matrix of pairwise similarities and we introduce a new technique, called "t-SNE", for visualizing the resulting similarity data. t-SNE is capable of capturing much of the local structure of the high-dimensional data very well, while also revealing global structure such as the presence of clusters at several scales. We illustrate the performance of t-SNE by comparing it to the seven dimensionality reduction techniques mentioned above on five data sets from a variety of domains. Because of space limitations, most of the $(7+1) \times 5 = 40$ maps are presented in the supplemental material, but the maps that we present in the paper are sufficient to demonstrate the superiority of t-SNE.

The outline of the paper is as follows. In Section 2, we outline SNE as presented by Hinton and Roweis (2002), which forms the basis for t-SNE. In Section 3, we present t-SNE, which has two important differences from SNE. In Section 4, we describe the experimental setup and the results of our experiments. Subsequently, Section 5 shows how t-SNE can be modified to visualize real-world data sets that contain many more than $10,000$ datapoints. The results of our experiments are discussed in more detail in Section 6. Our conclusions and suggestions for future work are presented in Section 7.

2. Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities.[1] The similarity of datapoint x_j to datapoint x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i. For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σ_i). Mathematically, the conditional probability $p_{j|i}$ is given by

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}, \tag{1}$$

where σ_i is the variance of the Gaussian that is centered on datapoint x_i. The method for determining the value of σ_i is presented later in this section. Because we are only interested in modeling pairwise similarities, we set the value of $p_{i|i}$ to zero. For the low-dimensional counterparts y_i and y_j of the high-dimensional datapoints x_i and x_j, it is possible to compute a similar conditional probability, which we denote by $q_{j|i}$. We set[2] the variance of the Gaussian that is employed in the computation of the conditional probabilities $q_{j|i}$ to $\frac{1}{\sqrt{2}}$. Hence, we model the similarity of map point y_j to map point y_i by

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k\neq i}\exp\left(-\|y_i - y_k\|^2\right)}.$$

Again, since we are only interested in modeling pairwise similarities, we set $q_{i|i} = 0$.

If the map points y_i and y_j correctly model the similarity between the high-dimensional datapoints x_i and x_j, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal. Motivated by this observation, SNE aims to find a low-dimensional data representation that minimizes the mismatch between $p_{j|i}$ and $q_{j|i}$. A natural measure of the faithfulness with which $q_{j|i}$ models $p_{j|i}$ is the Kullback-Leibler divergence (which is in this case equal to the cross-entropy up to an additive constant). SNE minimizes the sum of Kullback-Leibler divergences over all datapoints using a gradient descent method. The cost function C is given by

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}, \tag{2}$$

in which P_i represents the conditional probability distribution over all other datapoints given datapoint x_i, and Q_i represents the conditional probability distribution over all other map points given map point y_i. Because the Kullback-Leibler divergence is not symmetric, different types of error in the pairwise distances in the low-dimensional map are not weighted equally. In particular, there is a large cost for using widely separated map points to represent nearby datapoints (i.e., for using

1. SNE can also be applied to data sets that consist of pairwise similarities between objects rather than high-dimensional vector representations of each object, provided these simiarities can be interpreted as conditional probabilities. For example, human word association data consists of the probability of producing each possible word in response to a given word, as a result of which it is already in the form required by SNE.
2. Setting the variance in the low-dimensional Gaussians to another value only results in a rescaled version of the final map. Note that by using the same variance for every datapoint in the low-dimensional map, we lose the property that the data is a perfect model of itself if we embed it in a space of the same dimensionality, because in the high-dimensional space, we used a different variance σ_i in each Gaussian.

a small $q_{j|i}$ to model a large $p_{j|i}$), but there is only a small cost for using nearby map points to represent widely separated datapoints. This small cost comes from wasting some of the probability mass in the relevant Q distributions. In other words, the SNE cost function focuses on retaining the local structure of the data in the map (for reasonable values of the variance of the Gaussian in the high-dimensional space, σ_i).

The remaining parameter to be selected is the variance σ_i of the Gaussian that is centered over each high-dimensional datapoint, x_i. It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of σ_i induces a probability distribution, P_i, over all of the other datapoints. This distribution has an entropy which increases as σ_i increases. SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user.[3] The perplexity is defined as

$$Perp(P_i) = 2^{H(P_i)},$$

where $H(P_i)$ is the Shannon entropy of P_i measured in bits

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

The perplexity can be interpreted as a smooth measure of the effective number of neighbors. The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50.

The minimization of the cost function in Equation 2 is performed using a gradient descent method. The gradient has a surprisingly simple form

$$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

Physically, the gradient may be interpreted as the resultant force created by a set of springs between the map point y_i and all other map points y_j. All springs exert a force along the direction $(y_i - y_j)$. The spring between y_i and y_j repels or attracts the map points depending on whether the distance between the two in the map is too small or too large to represent the similarities between the two high-dimensional datapoints. The force exerted by the spring between y_i and y_j is proportional to its length, and also proportional to its stiffness, which is the mismatch $(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})$ between the pairwise similarities of the data points and the map points.

The gradient descent is initialized by sampling map points randomly from an isotropic Gaussian with small variance that is centered around the origin. In order to speed up the optimization and to avoid poor local minima, a relatively large momentum term is added to the gradient. In other words, the current gradient is added to an exponentially decaying sum of previous gradients in order to determine the changes in the coordinates of the map points at each iteration of the gradient search. Mathematically, the gradient update with a momentum term is given by

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

3. Note that the perplexity increases monotonically with the variance σ_i.

where $\mathcal{Y}^{(t)}$ indicates the solution at iteration t, η indicates the learning rate, and $\alpha(t)$ represents the momentum at iteration t.

In addition, in the early stages of the optimization, Gaussian noise is added to the map points after each iteration. Gradually reducing the variance of this noise performs a type of simulated annealing that helps the optimization to escape from poor local minima in the cost function. If the variance of the noise changes very slowly at the critical point at which the global structure of the map starts to form, SNE tends to find maps with a better global organization. Unfortunately, this requires sensible choices of the initial amount of Gaussian noise and the rate at which it decays. Moreover, these choices interact with the amount of momentum and the step size that are employed in the gradient descent. It is therefore common to run the optimization several times on a data set to find appropriate values for the parameters.[4] In this respect, SNE is inferior to methods that allow convex optimization and it would be useful to find an optimization method that gives good results without requiring the extra computation time and parameter choices introduced by the simulated annealing.

3. t-Distributed Stochastic Neighbor Embedding

Section 2 discussed SNE as it was presented by Hinton and Roweis (2002). Although SNE constructs reasonably good visualizations, it is hampered by a cost function that is difficult to optimize and by a problem we refer to as the "crowding problem". In this section, we present a new technique called "t-Distributed Stochastic Neighbor Embedding" or "t-SNE" that aims to alleviate these problems. The cost function used by t-SNE differs from the one used by SNE in two ways: (1) it uses a symmetrized version of the SNE cost function with simpler gradients that was briefly introduced by Cook et al. (2007) and (2) it uses a Student-t distribution rather than a Gaussian to compute the similarity between two points *in the low-dimensional space*. t-SNE employs a heavy-tailed distribution in the low-dimensional space to alleviate both the crowding problem and the optimization problems of SNE.

In this section, we first discuss the symmetric version of SNE (Section 3.1). Subsequently, we discuss the crowding problem (Section 3.2), and the use of heavy-tailed distributions to address this problem (Section 3.3). We conclude the section by describing our approach to the optimization of the t-SNE cost function (Section 3.4).

3.1 Symmetric SNE

As an alternative to minimizing the sum of the Kullback-Leibler divergences between the conditional probabilities $p_{j|i}$ and $q_{j|i}$, it is also possible to minimize a single Kullback-Leibler divergence between a joint probability distribution, P, in the high-dimensional space and a joint probability distribution, Q, in the low-dimensional space:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

where again, we set p_{ii} and q_{ii} to zero. We refer to this type of SNE as symmetric SNE, because it has the property that $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ for $\forall i, j$. In symmetric SNE, the pairwise similarities in

4. Picking the best map after several runs as a visualization of the data is not nearly as problematic as picking the model that does best on a test set during supervised learning. In visualization, the aim is to see the structure in the training data, not to generalize to held out test data.

the low-dimensional map q_{ij} are given by

$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)}, \tag{3}$$

The obvious way to define the pairwise similarities in the high-dimensional space p_{ij} is

$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2/2\sigma^2\right)},$$

but this causes problems when a high-dimensional datapoint x_i is an outlier (i.e., all pairwise distances $\|x_i - x_j\|^2$ are large for x_i). For such an outlier, the values of p_{ij} are extremely small for all j, so the location of its low-dimensional map point y_i has very little effect on the cost function. As a result, the position of the map point is not well determined by the positions of the other map points. We circumvent this problem by defining the joint probabilities p_{ij} in the high-dimensional space to be the symmetrized conditional probabilities, that is, we set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$. This ensures that $\sum_j p_{ij} > \frac{1}{2n}$ for all datapoints x_i, as a result of which each datapoint x_i makes a significant contribution to the cost function. In the low-dimensional space, symmetric SNE simply uses Equation 3. The main advantage of the symmetric version of SNE is the simpler form of its gradient, which is faster to compute. The gradient of symmetric SNE is fairly similar to that of asymmetric SNE, and is given by

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j).$$

In preliminary experiments, we observed that symmetric SNE seems to produce maps that are just as good as asymmetric SNE, and sometimes even a little better.

3.2 The Crowding Problem

Consider a set of datapoints that lie on a two-dimensional curved manifold which is approximately linear on a small scale, and which is embedded within a higher-dimensional space. It is possible to model the small pairwise distances between datapoints fairly well in a two-dimensional map, which is often illustrated on toy examples such as the "Swiss roll" data set. Now suppose that the manifold has ten intrinsic dimensions[5] and is embedded within a space of much higher dimensionality. There are several reasons why the pairwise distances in a two-dimensional map cannot faithfully model distances between points on the ten-dimensional manifold. For instance, in ten dimensions, it is possible to have 11 datapoints that are mutually equidistant and there is no way to model this faithfully in a two-dimensional map. A related problem is the very different distribution of pairwise distances in the two spaces. The volume of a sphere centered on datapoint i scales as r^m, where r is the radius and m the dimensionality of the sphere. So if the datapoints are approximately uniformly distributed in the region around i on the ten-dimensional manifold, and we try to model the distances from i to the other datapoints in the two-dimensional map, we get the following "crowding problem": the area of the two-dimensional map that is available to accommodate moderately distant datapoints will not be nearly large enough compared with the area available to accommodate nearby datapoints. Hence, if we want to model the small distances accurately in the map, most of the points

5. This is approximately correct for the images of handwritten digits we use in our experiments in Section 4.

that are at a moderate distance from datapoint i will have to be placed much too far away in the two-dimensional map. In SNE, the spring connecting datapoint i to each of these too-distant map points will thus exert a very small attractive force. Although these attractive forces are very small, the very large number of such forces crushes together the points in the center of the map, which prevents gaps from forming between the natural clusters. Note that the crowding problem is not specific to SNE, but that it also occurs in other local techniques for multidimensional scaling such as Sammon mapping.

An attempt to address the crowding problem by adding a slight repulsion to all springs was presented by Cook et al. (2007). The slight repulsion is created by introducing a uniform background model with a small mixing proportion, ρ. So however far apart two map points are, q_{ij} can never fall below $\frac{2\rho}{n(n-1)}$ (because the uniform background distribution is over $n(n-1)/2$ pairs). As a result, for datapoints that are far apart in the high-dimensional space, q_{ij} will always be larger than p_{ij}, leading to a slight repulsion. This technique is called UNI-SNE and although it usually outperforms standard SNE, the optimization of the UNI-SNE cost function is tedious. The best optimization method known is to start by setting the background mixing proportion to zero (i.e., by performing standard SNE). Once the SNE cost function has been optimized using simulated annealing, the background mixing proportion can be increased to allow some gaps to form between natural clusters as shown by Cook et al. (2007). Optimizing the UNI-SNE cost function directly does not work because two map points that are far apart will get almost all of their q_{ij} from the uniform background. So even if their p_{ij} is large, there will be no attractive force between them, because a small change in their separation will have a vanishingly small *proportional* effect on q_{ij}. This means that if two parts of a cluster get separated early on in the optimization, there is no force to pull them back together.

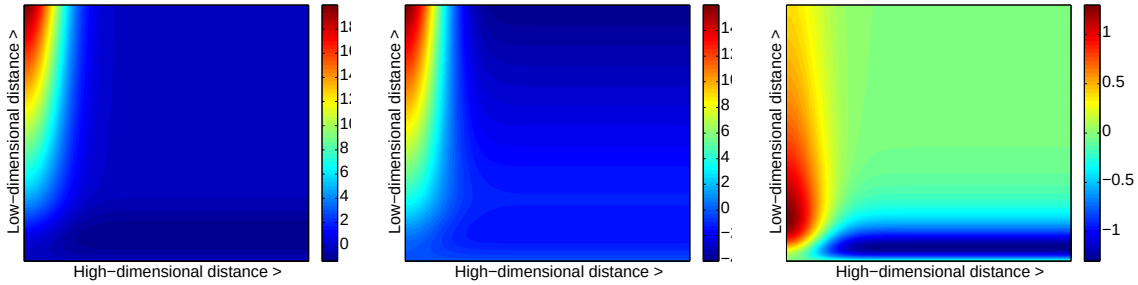
3.3 Mismatched Tails can Compensate for Mismatched Dimensionalities

Since symmetric SNE is actually matching the joint probabilities of pairs of datapoints in the high-dimensional and the low-dimensional spaces rather than their distances, we have a natural way of alleviating the crowding problem that works as follows. In the high-dimensional space, we convert distances into probabilities using a Gaussian distribution. In the low-dimensional map, we can use a probability distribution that has much heavier tails than a Gaussian to convert distances into probabilities. This allows a moderate distance in the high-dimensional space to be faithfully modeled by a much larger distance in the map and, as a result, it eliminates the unwanted attractive forces between map points that represent moderately dissimilar datapoints.

In t-SNE, we employ a Student t-distribution with one degree of freedom (which is the same as a Cauchy distribution) as the heavy-tailed distribution in the low-dimensional map. Using this distribution, the joint probabilities q_{ij} are defined as

$$q_{ij} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1+\|y_k - y_l\|^2\right)^{-1}}. \tag{4}$$

We use a Student t-distribution with a single degree of freedom, because it has the particularly nice property that $\left(1+\|y_i - y_j\|^2\right)^{-1}$ approaches an inverse square law for large pairwise distances $\|y_i - y_j\|$ in the low-dimensional map. This makes the map's representation of joint probabilities (almost) invariant to changes in the scale of the map for map points that are far apart. It also means that large clusters of points that are far apart interact in just the same way as individual points, so the optimization operates in the same way at all but the finest scales. A theoretical justification for our



(a) Gradient of SNE. (b) Gradient of UNI-SNE. (c) Gradient of t-SNE.

Figure 1: Gradients of three types of SNE as a function of the pairwise Euclidean distance between two points in the high-dimensional and the pairwise distance between the points in the low-dimensional data representation.

selection of the Student t-distribution is that it is closely related to the Gaussian distribution, as the Student t-distribution is an infinite mixture of Gaussians. A computationally convenient property is that it is much faster to evaluate the density of a point under a Student t-distribution than under a Gaussian because it does not involve an exponential, even though the Student t-distribution is equivalent to an infinite mixture of Gaussians with different variances.

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q (computed using Equation 4) is derived in Appendix A, and is given by

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1}. \tag{5}$$

In Figure 1(a) to 1(c), we show the gradients between two low-dimensional datapoints y_i and y_j as a function of their pairwise Euclidean distances in the high-dimensional and the low-dimensional space (i.e., as a function of $\|x_i - x_j\|$ and $\|y_i - y_j\|$) for the symmetric versions of SNE, UNI-SNE, and t-SNE. In the figures, positive values of the gradient represent an attraction between the low-dimensional datapoints y_i and y_j, whereas negative values represent a repulsion between the two datapoints. From the figures, we observe two main advantages of the t-SNE gradient over the gradients of SNE and UNI-SNE.

First, the t-SNE gradient strongly repels dissimilar datapoints that are modeled by a small pairwise distance in the low-dimensional representation. SNE has such a repulsion as well, but its effect is minimal compared to the strong attractions elsewhere in the gradient (the largest attraction in our graphical representation of the gradient is approximately 19, whereas the largest repulsion is approximately 1). In UNI-SNE, the amount of repulsion between dissimilar datapoints is slightly larger, however, this repulsion is only strong when the pairwise distance between the points in the low-dimensional representation is already large (which is often not the case, since the low-dimensional representation is initialized by sampling from a Gaussian with a very small variance that is centered around the origin).

Second, although t-SNE introduces strong repulsions between dissimilar datapoints that are modeled by small pairwise distances, these repulsions do not go to infinity. In this respect, t-SNE differs from UNI-SNE, in which the strength of the repulsion between very dissimilar datapoints

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

is proportional to their pairwise distance in the low-dimensional map, which may cause dissimilar datapoints to move much too far away from each other.

Taken together, t-SNE puts emphasis on (1) modeling dissimilar datapoints by means of large pairwise distances, and (2) modeling similar datapoints by means of small pairwise distances. Moreover, as a result of these characteristics of the t-SNE cost function (and as a result of the approximate scale invariance of the Student t-distribution), the optimization of the t-SNE cost function is much easier than the optimization of the cost functions of SNE and UNI-SNE. Specifically, t-SNE introduces long-range forces in the low-dimensional map that can pull back together two (clusters of) similar points that get separated early on in the optimization. SNE and UNI-SNE do not have such long-range forces, as a result of which SNE and UNI-SNE need to use simulated annealing to obtain reasonable solutions. Instead, the long-range forces in t-SNE facilitate the identification of good local optima without resorting to simulated annealing.

3.4 Optimization Methods for t-SNE

We start by presenting a relatively simple, gradient descent procedure for optimizing the t-SNE cost function. This simple procedure uses a momentum term to reduce the number of iterations required and it works best if the momentum term is small until the map points have become moderately well organized. Pseudocode for this simple algorithm is presented in Algorithm 1. The simple algorithm can be sped up using the adaptive learning rate scheme that is described by Jacobs (1988), which gradually increases the learning rate in directions in which the gradient is stable.

Although the simple algorithm produces visualizations that are often much better than those produced by other non-parametric dimensionality reduction techniques, the results can be improved further by using either of two tricks. The first trick, which we call "early compression", is to force the map points to stay close together at the start of the optimization. When the distances between map points are small, it is easy for clusters to move through one another so it is much easier to explore the space of possible global organizations of the data. Early compression is implemented by adding an additional L2-penalty to the cost function that is proportional to the sum of squared

distances of the map points from the origin. The magnitude of this penalty term and the iteration at which it is removed are set by hand, but the behavior is fairly robust across variations in these two additional optimization parameters.

A less obvious way to improve the optimization, which we call "early exaggeration", is to multiply all of the p_{ij}'s by, for example, 4, in the initial stages of the optimization. This means that almost all of the q_{ij}'s, which still add up to 1, are much too small to model their corresponding p_{ij}'s. As a result, the optimization is encouraged to focus on modeling the large p_{ij}'s by fairly large q_{ij}'s. The effect is that the natural clusters in the data tend to form tight widely separated clusters in the map. This creates a lot of relatively empty space in the map, which makes it much easier for the clusters to move around relative to one another in order to find a good global organization.

In all the visualizations presented in this paper and in the supporting material, we used exactly the same optimization procedure. We used the early exaggeration method with an exaggeration of 4 for the first 50 iterations (note that early exaggeration is not included in the pseudocode in Algorithm 1). The number of gradient descent iterations T was set 1000, and the momentum term was set to $\alpha^{(t)} = 0.5$ for $t < 250$ and $\alpha^{(t)} = 0.8$ for $t \geq 250$. The learning rate η is initially set to 100 and it is updated after every iteration by means of the adaptive learning rate scheme described by Jacobs (1988). A Matlab implementation of the resulting algorithm is available at `http://ticc.uvt.nl/~lvdrmaaten/tsne`.

4. Experiments

To evaluate t-SNE, we present experiments in which t-SNE is compared to seven other non-parametric techniques for dimensionality reduction. Because of space limitations, in the paper, we only compare t-SNE with: (1) Sammon mapping, (2) Isomap, and (3) LLE. In the supporting material, we also compare t-SNE with: (4) CCA, (5) SNE, (6) MVU, and (7) Laplacian Eigenmaps. We performed experiments on five data sets that represent a variety of application domains. Again because of space limitations, we restrict ourselves to three data sets in the paper. The results of our experiments on the remaining two data sets are presented in the supplemental material.

In Section 4.1, the data sets that we employed in our experiments are introduced. The setup of the experiments is presented in Section 4.2. In Section 4.3, we present the results of our experiments.

4.1 Data Sets

The five data sets we employed in our experiments are: (1) the MNIST data set, (2) the Olivetti faces data set, (3) the COIL-20 data set, (4) the word-features data set, and (5) the Netflix data set. We only present results on the first three data sets in this section. The results on the remaining two data sets are presented in the supporting material. The first three data sets are introduced below.

The MNIST data set[6] contains 60,000 grayscale images of handwritten digits. For our experiments, we randomly selected 6,000 of the images for computational reasons. The digit images have $28 \times 28 = 784$ pixels (i.e., dimensions). The Olivetti faces data set[7] consists of images of 40 individuals with small variations in viewpoint, large variations in expression, and occasional addition of glasses. The data set consists of 400 images (10 per individual) of size $92 \times 112 = 10,304$ pixels, and is labeled according to identity. The COIL-20 data set (Nene et al., 1996) contains images of 20

6. The MNIST data set is publicly available from `http://yann.lecun.com/exdb/mnist/index.html`.
7. The Olivetti faces data set is publicly available from `http://mambo.ucsc.edu/psl/olivetti.html`.

different objects viewed from 72 equally spaced orientations, yielding a total of 1,440 images. The images contain $32 \times 32 = 1,024$ pixels.

4.2 Experimental Setup

In all of our experiments, we start by using PCA to reduce the dimensionality of the data to 30. This speeds up the computation of pairwise distances between the datapoints and suppresses some noise without severely distorting the interpoint distances. We then use each of the dimensionality reduction techniques to convert the 30-dimensional representation to a two-dimensional map and we show the resulting map as a scatterplot. For all of the data sets, there is information about the class of each datapoint, but the class information is only used to select a color and/or symbol for the map points. The class information is not used to determine the spatial coordinates of the map points. The coloring thus provides a way of evaluating how well the map preserves the similarities within each class.

The cost function parameter settings we employed in our experiments are listed in Table 1. In the table, $Perp$ represents the perplexity of the conditional probability distribution induced by a Gaussian kernel and k represents the number of nearest neighbors employed in a neighborhood graph. In the experiments with Isomap and LLE, we only visualize datapoints that correspond to vertices in the largest connected component of the neighborhood graph.[8] For the Sammon mapping optimization, we performed Newton's method for 500 iterations.

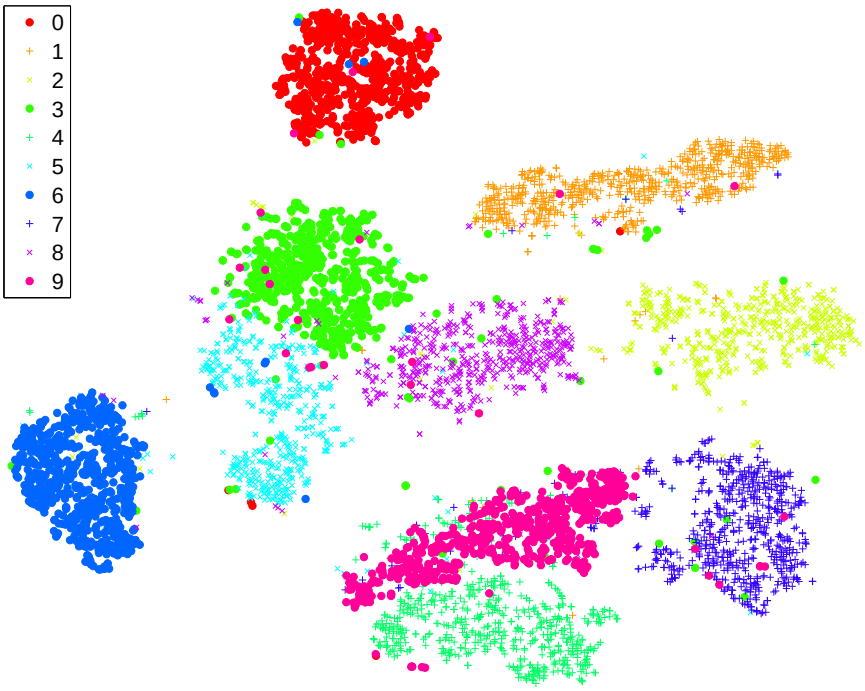
Technique	*Cost function parameters*
t-SNE	$Perp = 40$
Sammon mapping	none
Isomap	$k = 12$
LLE	$k = 12$

Table 1: Cost function parameter settings for the experiments.

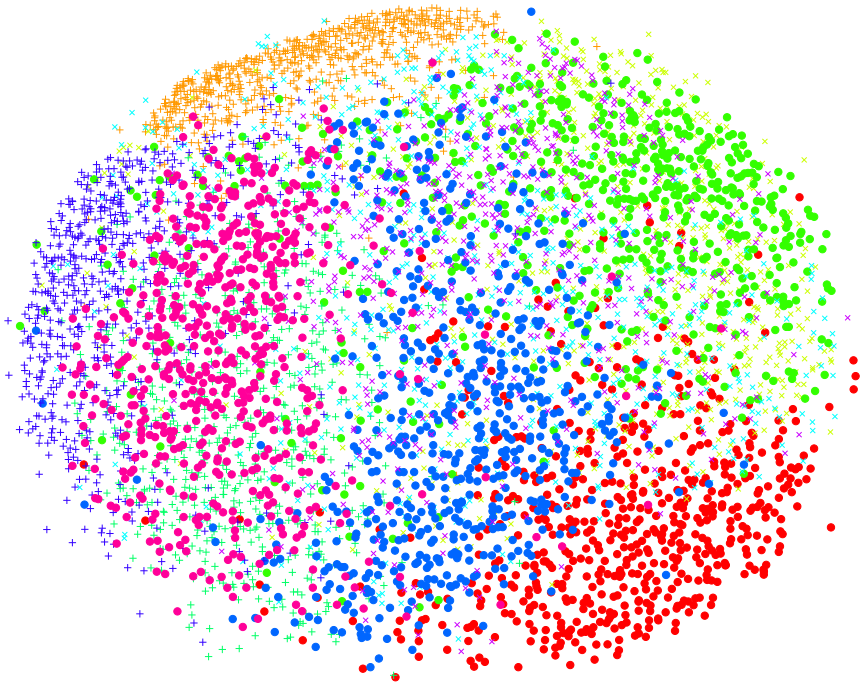
4.3 Results

In Figures 2 and 3, we show the results of our experiments with t-SNE, Sammon mapping, Isomap, and LLE on the MNIST data set. The results reveal the strong performance of t-SNE compared to the other techniques. In particular, Sammon mapping constructs a "ball" in which only three classes (representing the digits 0, 1, and 7) are somewhat separated from the other classes. Isomap and LLE produce solutions in which there are large overlaps between the digit classes. In contrast, t-SNE constructs a map in which the separation between the digit classes is almost perfect. Moreover, detailed inspection of the t-SNE map reveals that much of the local structure of the data (such as the orientation of the ones) is captured as well. This is illustrated in more detail in Section 5 (see Figure 7). The map produced by t-SNE contains some points that are clustered with the wrong class, but most of these points correspond to distorted digits many of which are difficult to identify.
Figure 4 shows the results of applying t-SNE, Sammon mapping, Isomap, and LLE to the Olivetti faces data set. Again, Isomap and LLE produce solutions that provide little insight into the class

8. Isomap and LLE require data that gives rise to a neighborhood graph that is connected.

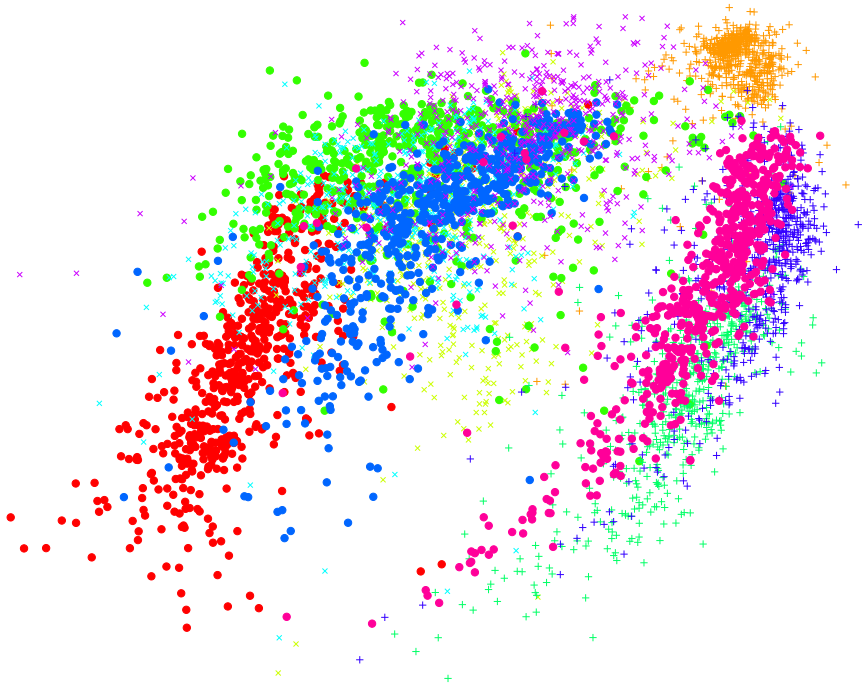


(a) Visualization by t-SNE.

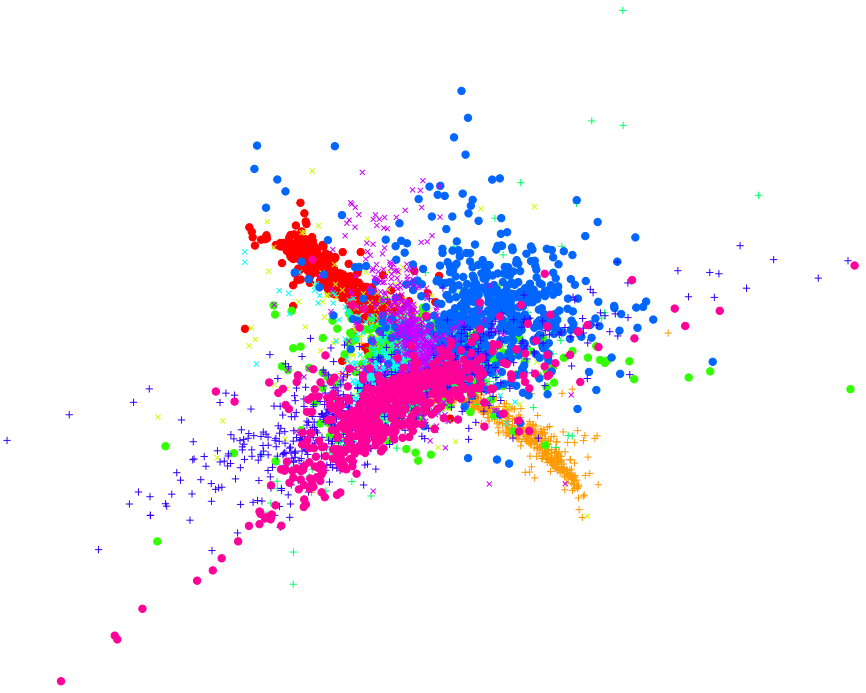


(b) Visualization by Sammon mapping.

Figure 2: Visualizations of 6,000 handwritten digits from the MNIST data set.

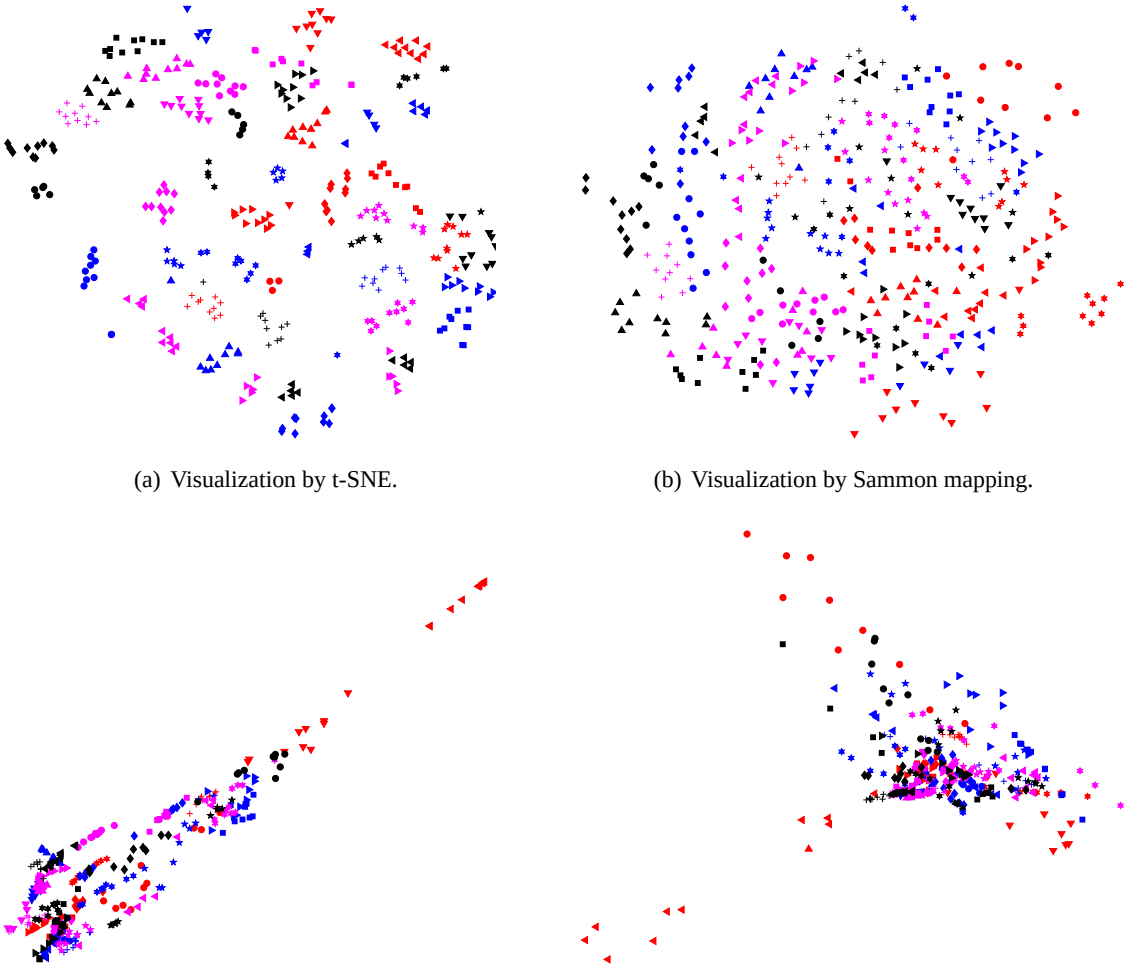


(a) Visualization by Isomap.



(b) Visualization by LLE.

Figure 3: Visualizations of 6,000 handwritten digits from the MNIST data set.



(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

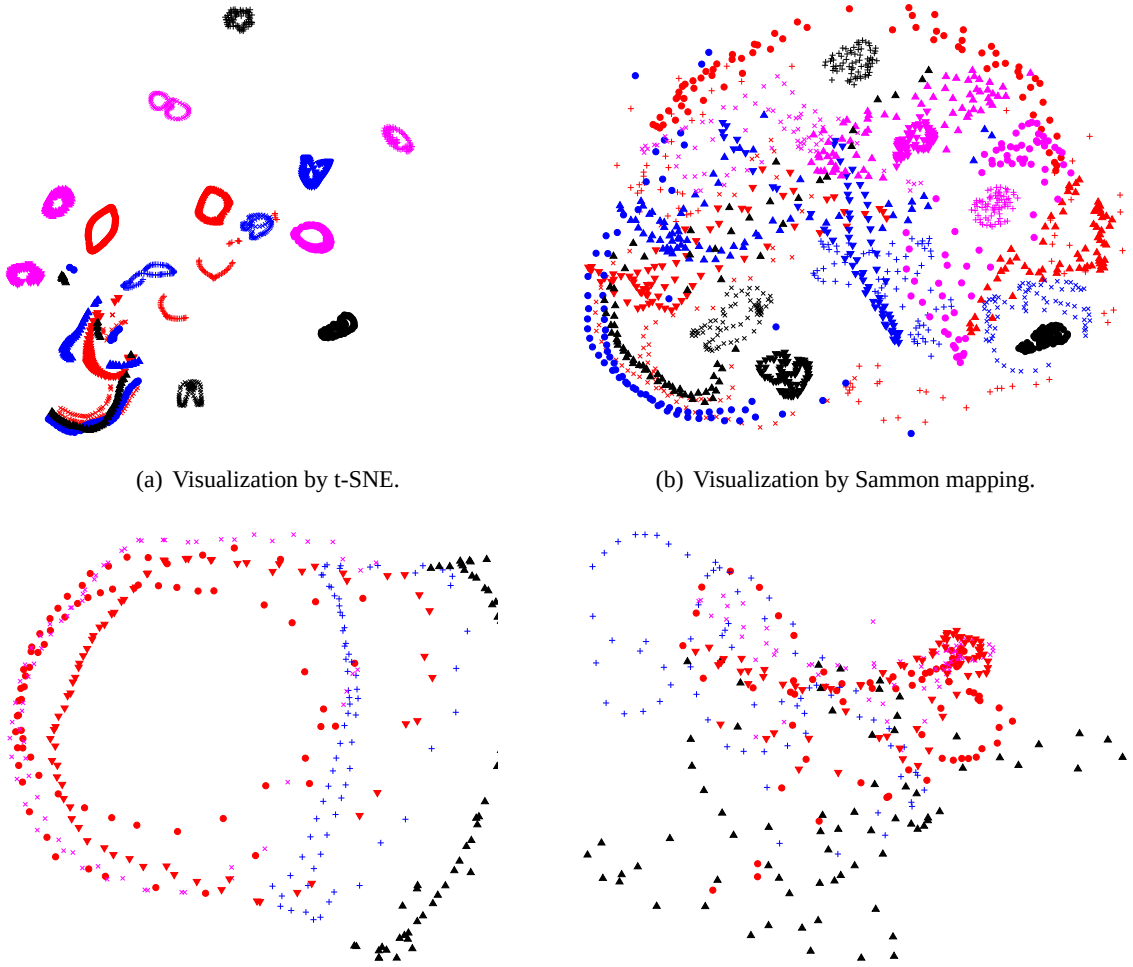
(c) Visualization by Isomap.

(d) Visualization by LLE.

Figure 4: Visualizations of the Olivetti faces data set.

structure of the data. The map constructed by Sammon mapping is significantly better, since it models many of the members of each class fairly close together, but none of the classes are clearly separated in the Sammon map. In contrast, t-SNE does a much better job of revealing the natural classes in the data. Some individuals have their ten images split into two clusters, usually because a subset of the images have the head facing in a significantly different direction, or because they have a very different expression or glasses. For these individuals, it is not clear that their ten images form a natural class when using Euclidean distance in pixel space.

Figure 5 shows the results of applying t-SNE, Sammon mapping, Isomap, and LLE to the COIL-20 data set. For many of the 20 objects, t-SNE accurately represents the one-dimensional manifold of viewpoints as a closed loop. For objects which look similar from the front and the back, t-SNE distorts the loop so that the images of front and back are mapped to nearby points. For the four types of toy car in the COIL-20 data set (the four aligned "sausages" in the bottom-left of the t-SNE map), the four rotation manifolds are aligned by the orientation of the cars to capture the high



(a) Visualization by t-SNE.

(b) Visualization by Sammon mapping.

(c) Visualization by Isomap.

(d) Visualization by LLE.

Figure 5: Visualizations of the COIL-20 data set.

similarity between different cars at the same orientation. This prevents t-SNE from keeping the four manifolds clearly separate. Figure 5 also reveals that the other three techniques are not nearly as good at cleanly separating the manifolds that correspond to very different objects. In addition, Isomap and LLE only visualize a small number of classes from the COIL-20 data set, because the data set comprises a large number of widely separated submanifolds that give rise to small connected components in the neighborhood graph.

5. Applying t-SNE to Large Data Sets

Like many other visualization techniques, t-SNE has a computational and memory complexity that is quadratic in the number of datapoints. This makes it infeasible to apply the standard version of t-SNE to data sets that contain many more than, say, 10,000 points. Obviously, it is possible to pick a random subset of the datapoints and display them using t-SNE, but such an approach fails to

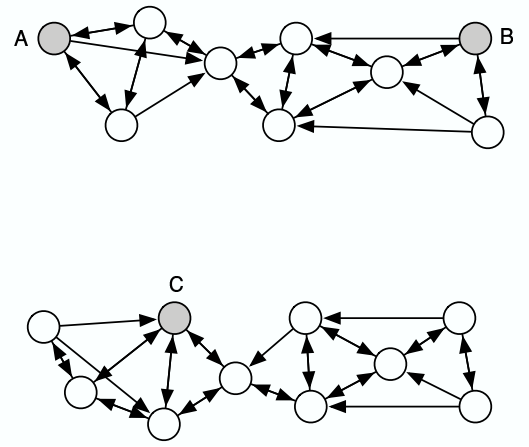


Figure 6: An illustration of the advantage of the random walk version of t-SNE over a standard landmark approach. The shaded points A, B, and C are three (almost) equidistant landmark points, whereas the non-shaded datapoints are non-landmark points. The arrows represent a directed neighborhood graph where $k = 3$. In a standard landmark approach, the pairwise affinity between A and B is approximately equal to the pairwise affinity between A and C. In the random walk version of t-SNE, the pairwise affinity between A and B is much larger than the pairwise affinity between A and C, and therefore, it reflects the structure of the data much better.

make use of the information that the undisplayed datapoints provide about the underlying manifolds. Suppose, for example, that A, B, and C are all equidistant in the high-dimensional space. If there are many undisplayed datapoints between A and B and none between A and C, it is much more likely that A and B are part of the same cluster than A and C. This is illustrated in Figure 6. In this section, we show how t-SNE can be modified to display a random subset of the datapoints (so-called landmark points) in a way that uses information from the entire (possibly very large) data set.

We start by choosing a desired number of neighbors and creating a neighborhood graph for all of the datapoints. Although this is computationally intensive, it is only done once. Then, for each of the landmark points, we define a random walk starting at that landmark point and terminating as soon as it lands on another landmark point. During a random walk, the probability of choosing an edge emanating from node x_i to node x_j is proportional to $e^{-\|x_i - x_j\|^2}$. We define $p_{j|i}$ to be the fraction of random walks starting at landmark point x_i that terminate at landmark point x_j. This has some resemblance to the way Isomap measures pairwise distances between points. However, as in diffusion maps (Lafon and Lee, 2006; Nadler et al., 2006), rather than looking for the shortest path through the neighborhood graph, the random walk-based affinity measure integrates over all paths through the neighborhood graph. As a result, the random walk-based affinity measure is much less sensitive to "short-circuits" (Lee and Verleysen, 2005), in which a single noisy datapoint provides a bridge between two regions of dataspace that should be far apart in the map. Similar approaches using random walks have also been successfully applied to, for example, semi-supervised learning (Szummer and Jaakkola, 2001; Zhu et al., 2003) and image segmentation (Grady, 2006).

The most obvious way to compute the random walk-based similarities $p_{j|i}$ is to explicitly perform the random walks on the neighborhood graph, which works very well in practice, given that one can easily perform one million random walks per second. Alternatively, Grady (2006) presents an analytical solution to compute the pairwise similarities $p_{j|i}$ that involves solving a sparse linear system. The analytical solution to compute the similarities $p_{j|i}$ is sketched in Appendix B. In preliminary experiments, we did not find significant differences between performing the random walks explicitly and the analytical solution. In the experiment we present below, we explicitly performed the random walks because this is computationally less expensive. However, for very large data sets in which the landmark points are very sparse, the analytical solution may be more appropriate.

Figure 7 shows the results of an experiment, in which we applied the random walk version of t-SNE to 6,000 randomly selected digits from the MNIST data set, using all 60,000 digits to compute the pairwise affinities $p_{j|i}$. In the experiment, we used a neighborhood graph that was constructed using a value of $k = 20$ nearest neighbors.[9] The inset of the figure shows the same visualization as a scatterplot in which the colors represent the labels of the digits. In the t-SNE map, all classes are clearly separated and the "continental" sevens form a small separate cluster. Moreover, t-SNE reveals the main dimensions of variation within each class, such as the orientation of the ones, fours, sevens, and nines, or the "loopiness" of the twos. The strong performance of t-SNE is also reflected in the generalization error of nearest neighbor classifiers that are trained on the low-dimensional representation. Whereas the generalization error (measured using 10-fold cross validation) of a 1-nearest neighbor classifier trained on the original 784-dimensional datapoints is 5.75%, the generalization error of a 1-nearest neighbor classifier trained on the two-dimensional data representation produced by t-SNE is only 5.13%. The computational requirements of random walk t-SNE are reasonable: it took only one hour of CPU time to construct the map in Figure 7.

6. Discussion

The results in the previous two sections (and those in the supplemental material) demonstrate the performance of t-SNE on a wide variety of data sets. In this section, we discuss the differences between t-SNE and other non-parametric techniques (Section 6.1), and we also discuss a number of weaknesses and possible improvements of t-SNE (Section 6.2).

6.1 Comparison with Related Techniques

Classical scaling (Torgerson, 1952), which is closely related to PCA (Mardia et al., 1979; Williams, 2002), finds a linear transformation of the data that minimizes the sum of the squared errors between high-dimensional pairwise distances and their low-dimensional representatives. A linear method such as classical scaling is not good at modeling curved manifolds and it focuses on preserving the distances between widely separated datapoints rather than on preserving the distances between nearby datapoints. An important approach that attempts to address the problems of classical scaling is the Sammon mapping (Sammon, 1969) which alters the cost function of classical scaling by dividing the squared error in the representation of each pairwise Euclidean distance by the original Euclidean distance in the high-dimensional space. The resulting cost function is given by

$$C = \frac{1}{\sum_{ij} \|x_i - x_j\|} \sum_{i \neq j} \frac{(\|x_i - x_j\| - \|y_i - y_j\|)^2}{\|x_i - x_j\|},$$

9. In preliminary experiments, we found the performance of random walk t-SNE to be very robust under changes of k.

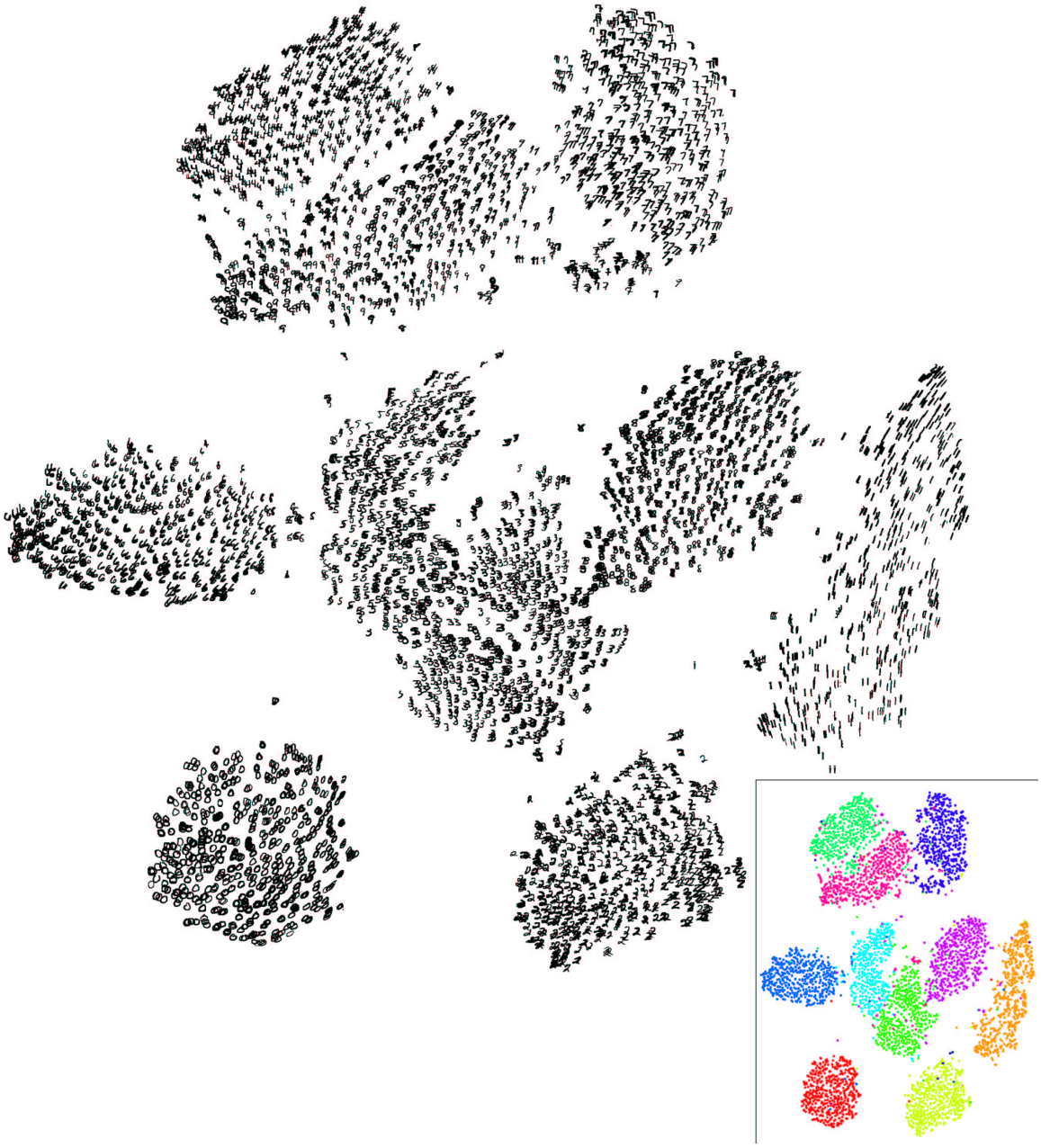


Figure 7: Visualization of 6,000 digits from the MNIST data set produced by the random walk version of t-SNE (employing all 60,000 digit images).

where the constant outside of the sum is added in order to simplify the derivation of the gradient. The main weakness of the Sammon cost function is that the importance of retaining small pairwise distances in the map is largely dependent on small differences in these pairwise distances. In particular, a small error in the model of two high-dimensional points that are extremely close together results in a large contribution to the cost function. Since all small pairwise distances constitute the local structure of the data, it seems more appropriate to aim to assign approximately equal importance to all small pairwise distances.

In contrast to Sammon mapping, the Gaussian kernel employed in the high-dimensional space by t-SNE defines a soft border between the local and global structure of the data and for pairs of datapoints that are close together relative to the standard deviation of the Gaussian, the importance of modeling their separations is almost independent of the magnitudes of those separations. Moreover, t-SNE determines the local neighborhood size for each datapoint separately based on the local density of the data (by forcing each conditional probability distribution P_i to have the same perplexity).

The strong performance of t-SNE compared to Isomap is partly explained by Isomap's susceptibility to "short-circuiting". Also, Isomap mainly focuses on modeling large geodesic distances rather than small ones.

The strong performance of t-SNE compared to LLE is mainly due to a basic weakness of LLE: the only thing that prevents all datapoints from collapsing onto a single point is a constraint on the covariance of the low-dimensional representation. In practice, this constraint is often satisfied by placing most of the map points near the center of the map and using a few widely scattered points to create large covariance (see Figure 3(b) and 4(d)). For neighborhood graphs that are almost disconnected, the covariance constraint can also be satisfied by a "curdled" map in which there are a few widely separated, collapsed subsets corresponding to the almost disconnected components. Furthermore, neighborhood-graph based techniques (such as Isomap and LLE) are not capable of visualizing data that consists of two or more widely separated submanifolds, because such data does not give rise to a connected neighborhood graph. It is possible to produce a separate map for each connected component, but this loses information about the relative similarities of the separate components.

Like Isomap and LLE, the random walk version of t-SNE employs neighborhood graphs, but it does not suffer from short-circuiting problems because the pairwise similarities between the high-dimensional datapoints are computed by integrating over all paths through the neighborhood graph. Because of the diffusion-based interpretation of the conditional probabilities underlying the random walk version of t-SNE, it is useful to compare t-SNE to diffusion maps. Diffusion maps define a "diffusion distance" on the high-dimensional datapoints that is given by

$$D^{(t)}(x_i, x_j) = \sqrt{\sum_k \frac{\left(p_{ik}^{(t)} - p_{jk}^{(t)}\right)^2}{\psi(x_k)^{(0)}}},$$

where $p_{ij}^{(t)}$ represents the probability of a particle traveling from x_i to x_j in t timesteps through a graph on the data with Gaussian emission probabilities. The term $\psi(x_k)^{(0)}$ is a measure for the local density of the points, and serves a similar purpose to the fixed perplexity Gaussian kernel that is employed in SNE. The diffusion map is formed by the principal non-trivial eigenvectors of the Markov matrix of the random walks of length t. It can be shown that when all $(n-1)$ non-trivial eigenvec-

tors are employed, the Euclidean distances in the diffusion map are equal to the diffusion distances in the high-dimensional data representation (Lafon and Lee, 2006). Mathematically, diffusion maps minimize

$$C = \sum_i \sum_j \left(D^{(t)}(x_i, x_j) - \|y_i - y_j\| \right)^2.$$

As a result, diffusion maps are susceptible to the same problems as classical scaling: they assign much higher importance to modeling the large pairwise diffusion distances than the small ones and as a result, they are not good at retaining the local structure of the data. Moreover, in contrast to the random walk version of t-SNE, diffusion maps do not have a natural way of selecting the length, t, of the random walks.

In the supplemental material, we present results that reveal that t-SNE outperforms CCA (Demartines and Hérault, 1997), MVU (Weinberger et al., 2004), and Laplacian Eigenmaps (Belkin and Niyogi, 2002) as well. For CCA and the closely related CDA (Lee et al., 2000), these results can be partially explained by the hard border λ that these techniques define between local and global structure, as opposed to the soft border of t-SNE. Moreover, within the range λ, CCA suffers from the same weakness as Sammon mapping: it assigns extremely high importance to modeling the distance between two datapoints that are extremely close.

Like t-SNE, MVU (Weinberger et al., 2004) tries to model all of the small separations well but MVU insists on modeling them perfectly (i.e., it treats them as constraints) and a single erroneous constraint may severely affect the performance of MVU. This can occur when there is a short-circuit between two parts of a curved manifold that are far apart in the intrinsic manifold coordinates. Also, MVU makes no attempt to model longer range structure: It simply pulls the map points as far apart as possible subject to the hard constraints so, unlike t-SNE, it cannot be expected to produce sensible large-scale structure in the map.

For Laplacian Eigenmaps, the poor results relative to t-SNE may be explained by the fact that Laplacian Eigenmaps have the same covariance constraint as LLE, and it is easy to cheat on this constraint.

6.2 Weaknesses

Although we have shown that t-SNE compares favorably to other techniques for data visualization, t-SNE has three potential weaknesses: (1) it is unclear how t-SNE performs on general dimensionality reduction tasks, (2) the relatively local nature of t-SNE makes it sensitive to the curse of the intrinsic dimensionality of the data, and (3) t-SNE is not guaranteed to converge to a global optimum of its cost function. Below, we discuss the three weaknesses in more detail.

1) Dimensionality reduction for other purposes. It is not obvious how t-SNE will perform on the more general task of dimensionality reduction (i.e., when the dimensionality of the data is not reduced to two or three, but to $d > 3$ dimensions). To simplify evaluation issues, this paper only considers the use of t-SNE for data visualization. The behavior of t-SNE when reducing data to two or three dimensions cannot readily be extrapolated to $d > 3$ dimensions because of the heavy tails of the Student-t distribution. In high-dimensional spaces, the heavy tails comprise a relatively large portion of the probability mass under the Student-t distribution, which might lead to d-dimensional data representations that do not preserve the local structure of the data as well. Hence, for tasks

in which the dimensionality of the data needs to be reduced to a dimensionality higher than three, Student t-distributions with more than one degree of freedom[10] are likely to be more appropriate.

2) Curse of intrinsic dimensionality. t-SNE reduces the dimensionality of data mainly based on local properties of the data, which makes t-SNE sensitive to the curse of the intrinsic dimensionality of the data (Bengio, 2007). In data sets with a high intrinsic dimensionality and an underlying manifold that is highly varying, the local linearity assumption on the manifold that t-SNE implicitly makes (by employing Euclidean distances between near neighbors) may be violated. As a result, t-SNE might be less successful if it is applied on data sets with a very high intrinsic dimensionality (for instance, a recent study by Meytlis and Sirovich (2007) estimates the space of images of faces to be constituted of approximately 100 dimensions). Manifold learners such as Isomap and LLE suffer from exactly the same problems (see, e.g., Bengio, 2007; van der Maaten et al., 2008). A possible way to (partially) address this issue is by performing t-SNE on a data representation obtained from a model that represents the highly varying data manifold efficiently in a number of nonlinear layers such as an autoencoder (Hinton and Salakhutdinov, 2006). Such deep-layer architectures can represent complex nonlinear functions in a much simpler way, and as a result, require fewer datapoints to learn an appropriate solution (as is illustrated for a d-bits parity task by Bengio 2007). Performing t-SNE on a data representation produced by, for example, an autoencoder is likely to improve the quality of the constructed visualizations, because autoencoders can identify highly-varying manifolds better than a local method such as t-SNE. However, the reader should note that it is by definition impossible to fully represent the structure of intrinsically high-dimensional data in two or three dimensions.

3) Non-convexity of the t-SNE cost function. A nice property of most state-of-the-art dimensionality reduction techniques (such as classical scaling, Isomap, LLE, and diffusion maps) is the convexity of their cost functions. A major weakness of t-SNE is that the cost function is not convex, as a result of which several optimization parameters need to be chosen. The constructed solutions depend on these choices of optimization parameters and may be different each time t-SNE is run from an initial random configuration of map points. We have demonstrated that the same choice of optimization parameters can be used for a variety of different visualization tasks, and we found that the quality of the optima does not vary much from run to run. Therefore, we think that the weakness of the optimization method is insufficient reason to reject t-SNE in favor of methods that lead to convex optimization problems but produce noticeably worse visualizations. A local optimum of a cost function that accurately captures what we want in a visualization is often preferable to the global optimum of a cost function that fails to capture important aspects of what we want. Moreover, the convexity of cost functions can be misleading, because their optimization is often computationally infeasible for large real-world data sets, prompting the use of approximation techniques (de Silva and Tenenbaum, 2003; Weinberger et al., 2007). Even for LLE and Laplacian Eigenmaps, the optimization is performed using iterative Arnoldi (Arnoldi, 1951) or Jacobi-Davidson (Fokkema et al., 1999) methods, which may fail to find the global optimum due to convergence problems.

7. Conclusions

The paper presents a new technique for the visualization of similarity data that is capable of retaining the local structure of the data while also revealing some important global structure (such as clusters

10. Increasing the degrees of freedom of a Student-t distribution makes the tails of the distribution lighter. With infinite degrees of freedom, the Student-t distribution is equal to the Gaussian distribution.

at multiple scales). Both the computational and the memory complexity of t-SNE are $O(n^2)$, but we present a landmark approach that makes it possible to successfully visualize large real-world data sets with limited computational demands. Our experiments on a variety of data sets show that t-SNE outperforms existing state-of-the-art techniques for visualizing a variety of real-world data sets. Matlab implementations of both the normal and the random walk version of t-SNE are available for download at `http://ticc.uvt.nl/~lvdrmaaten/tsne`.

In future work we plan to investigate the optimization of the number of degrees of freedom of the Student-t distribution used in t-SNE. This may be helpful for dimensionality reduction when the low-dimensional representation has many dimensions. We will also investigate the extension of t-SNE to models in which each high-dimensional datapoint is modeled by several low-dimensional map points as in Cook et al. (2007). Also, we aim to develop a parametric version of t-SNE that allows for generalization to held-out test data by using the t-SNE objective function to train a multilayer neural network that provides an explicit mapping to the low-dimensional space.

Acknowledgments

The authors thank Sam Roweis for many helpful discussions, Andriy Mnih for supplying the word-features data set, Ruslan Salakhutdinov for help with the Netflix data set (results for these data sets are presented in the supplemental material), and Guido de Croon for pointing us to the analytical solution of the random walk probabilities.

Laurens van der Maaten is supported by the CATCH-programme of the Dutch Scientific Organization (NWO), project RICH (grant 640.002.401), and cooperates with RACM. Geoffrey Hinton is a fellow of the Canadian Institute for Advanced Research, and is also supported by grants from NSERC and CFI and gifts from Google and Microsoft.

Appendix A. Derivation of the t-SNE gradient

t-SNE minimizes the Kullback-Leibler divergence between the joint probabilities p_{ij} in the high-dimensional space and the joint probabilities q_{ij} in the low-dimensional space. The values of p_{ij} are defined to be the symmetrized conditional probabilities, whereas the values of q_{ij} are obtained by means of a Student-t distribution with one degree of freedom

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n},$$

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}},$$

where $p_{j|i}$ and $p_{i|j}$ are either obtained from Equation 1 or from the random walk procedure described in Section 5. The values of p_{ii} and q_{ii} are set to zero. The Kullback-Leibler divergence between the two joint probability distributions P and Q is given by

$$\begin{aligned} C = KL(P||Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij}. \end{aligned} \tag{6}$$

In order to make the derivation less cluttered, we define two auxiliary variables d_{ij} and Z as follows

$$d_{ij} = \|y_i - y_j\|,$$
$$Z = \sum_{k \neq l} (1 + d_{kl}^2)^{-1}.$$

Note that if y_i changes, the only pairwise distances that change are d_{ij} and d_{ji} for $\forall j$. Hence, the gradient of the cost function C with respect to y_i is given by

$$\begin{aligned} \frac{\delta C}{\delta y_i} &= \sum_j \left(\frac{\delta C}{\delta d_{ij}} + \frac{\delta C}{\delta d_{ji}} \right) (y_i - y_j) \\ &= 2 \sum_j \frac{\delta C}{\delta d_{ij}} (y_i - y_j). \end{aligned} \tag{7}$$

The gradient $\frac{\delta C}{\delta d_{ij}}$ is computed from the definition of the Kullback-Leibler divergence in Equation 6 (note that the first part of this equation is a constant).

$$\begin{aligned} \frac{\delta C}{\delta d_{ij}} &= -\sum_{k \neq l} p_{kl} \frac{\delta (\log q_{kl})}{\delta d_{ij}} \\ &= -\sum_{k \neq l} p_{kl} \frac{\delta (\log q_{kl} Z - \log Z)}{\delta d_{ij}} \\ &= -\sum_{k \neq l} p_{kl} \left(\frac{1}{q_{kl} Z} \frac{\delta ((1 + d_{kl}^2)^{-1})}{\delta d_{ij}} - \frac{1}{Z} \frac{\delta Z}{\delta d_{ij}} \right) \end{aligned}$$

The gradient $\frac{\delta ((1+d_{kl}^2)^{-1})}{\delta d_{ij}}$ is only nonzero when $k = i$ and $l = j$. Hence, the gradient $\frac{\delta C}{\delta d_{ij}}$ is given by

$$\frac{\delta C}{\delta d_{ij}} = 2 \frac{p_{ij}}{q_{ij} Z} (1 + d_{ij}^2)^{-2} - 2 \sum_{k \neq l} p_{kl} \frac{(1 + d_{ij}^2)^{-2}}{Z}.$$

Noting that $\sum_{k \neq l} p_{kl} = 1$, we see that the gradient simplifies to

$$\begin{aligned} \frac{\delta C}{\delta d_{ij}} &= 2 p_{ij} (1 + d_{ij}^2)^{-1} - 2 q_{ij} (1 + d_{ij}^2)^{-1} \\ &= 2 (p_{ij} - q_{ij}) (1 + d_{ij}^2)^{-1}. \end{aligned}$$

Substituting this term into Equation 7, we obtain the gradient

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij}) (1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j).$$

Appendix B. Analytical Solution to Random Walk Probabilities

Below, we describe the analytical solution to the random walk probabilities that are employed in the random walk version of t-SNE (see Section 5). The solution is described in more detail by Grady (2006).

It can be shown that computing the probability that a random walk initiated from a non-landmark point (on a graph that is specified by adjacency matrix W) first reaches a specific landmark point is equal to computing the solution to the combinatorial Dirichlet problem in which the boundary conditions are at the locations of the landmark points, the considered landmark point is fixed to unity, and the other landmarks points are set to zero (Kakutani, 1945; Doyle and Snell, 1984). In practice, the solution can thus be obtained by minimizing the combinatorial formulation of the Dirichlet integral

$$D[x] = \frac{1}{2} x^T L x,$$

where L represents the graph Laplacian. Mathematically, the graph Laplacian is given by $L = D - W$, where $D = \text{diag}\left(\sum_j w_{1j}, \sum_j w_{2j}, ..., \sum_j w_{nj}\right)$. Without loss of generality, we may reorder the landmark points such that the landmark points come first. As a result, the combinatorial Dirichlet integral decomposes into

$$\begin{aligned} D[x_N] &= \frac{1}{2} \begin{bmatrix} x_L^T & x_N^T \end{bmatrix} \begin{bmatrix} L_L & B \\ B^T & L_N \end{bmatrix} \begin{bmatrix} x_L \\ x_N \end{bmatrix} \\ &= \frac{1}{2} \left(x_L^T L_L x_L + 2 x_N^T B^T x_M + x_N^T L_N x_N \right), \end{aligned}$$

where we use the subscript $\cdot_L$ to indicate the landmark points, and the subscript $\cdot_N$ to indicate the non-landmark points. Differentiating $D[x_N]$ with respect to x_N and finding its critical points amounts to solving the linear systems

$$L_N x_N = -B^T. \tag{8}$$

Please note that in this linear system, B^T is a matrix containing the columns from the graph Laplacian L that correspond to the landmark points (excluding the rows that correspond to landmark points). After normalization of the solutions to the systems X_N, the column vectors of X_N contain the probability that a random walk initiated from a non-landmark point terminates in a landmark point. One should note that the linear system in Equation 8 is only nonsingular if the graph is completely connected, or if each connected component in the graph contains at least one landmark point (Biggs, 1974).

Because we are interested in the probability of a random walk initiated from a *landmark point* terminating at another landmark point, we duplicate all landmark points in the neighborhood graph, and initiate the random walks from the duplicate landmarks. Because of memory constraints, it is not possible to store the entire matrix X_N into memory (note that we are only interested in a small number of rows from this matrix, viz., in the rows corresponding to the duplicate landmark points). Hence, we solve the linear systems defined by the columns of $-B^T$ one-by-one, and store only the parts of the solutions that correspond to the duplicate landmark points. For computational reasons, we first perform a Cholesky factorization of L_N, such that $L_N = CC^T$, where C is an upper-triangular matrix. Subsequently, the solution to the linear system in Equation 8 is obtained by solving the linear systems $Cy = -B^T$ and $Cx_N = y$ using a fast backsubstitution method.

References

W.E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–25, 1951.

G.D. Battista, P. Eades, R. Tamassia, and I.G. Tollis. Annotated bibliography on graph drawing. *Computational Geometry: Theory and Applications*, 4:235–282, 1994.

M. Belkin and P. Niyogi. Laplacian Eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14, pages 585–591, Cambridge, MA, USA, 2002. The MIT Press.

Y. Bengio. Learning deep architectures for AI. Technical Report 1312, Université de Montréal, 2007.

N. Biggs. Algebraic graph theory. In *Cambridge Tracts in Mathematics*, volume 67. Cambridge University Press, 1974.

H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

J.A. Cook, I. Sutskever, A. Mnih, and G.E. Hinton. Visualizing similarity data with a mixture of maps. In *Proceedings of the* 11^{th} *International Conference on Artificial Intelligence and Statistics*, volume 2, pages 67–74, 2007.

M.C. Ferreira de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394, 2003.

V. de Silva and J.B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 721–728, Cambridge, MA, USA, 2003. The MIT Press.

P. Demartines and J. Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, 1997.

P. Doyle and L. Snell. Random walks and electric networks. In *Carus mathematical monographs*, volume 22. Mathematical Association of America, 1984.

D.R. Fokkema, G.L.G. Sleijpen, and H.A. van der Vorst. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing*, 20(1):94–125, 1999.

L. Grady. Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1768–1783, 2006.

G.E. Hinton and S.T. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 15, pages 833–840, Cambridge, MA, USA, 2002. The MIT Press.

G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1: 295–307, 1988.

S. Kakutani. Markov processes and the Dirichlet problem. *Proceedings of the Japan Academy*, 21: 227–233, 1945.

D.A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.

S. Lafon and A.B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006.

J.A. Lee and M. Verleysen. Nonlinear dimensionality reduction of data manifolds with essential loops. *Neurocomputing*, 67:29–53, 2005.

J.A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, New York, NY, USA, 2007.

J.A. Lee, A. Lendasse, N. Donckers, and M. Verleysen. A robust nonlinear projection method. In *Proceedings of the* 8^{th} *European Symposium on Artificial Neural Networks*, pages 13–20, 2000.

K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic Press, 1979.

M. Meytlis and L. Sirovich. On the dimensionality of face space. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 29(7):1262–1267, 2007.

B. Nadler, S. Lafon, R.R. Coifman, and I.G. Kevrekidis. Diffusion maps, spectral clustering and the reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis: Special Issue on Diffusion Maps and Wavelets*, 21:113–127, 2006.

S.A. Nene, S.K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96, Columbia University, 1996.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969.

L. Song, A.J. Smola, K. Borgwardt, and A. Gretton. Colored Maximum Variance Unfolding. In *Advances in Neural Information Processing Systems*, volume 21 (in press), 2007.

W.N. Street, W.H. Wolberg, and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Proceedings of the IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology*, volume 1905, pages 861–870, 1993.

M. Szummer and T. Jaakkola. Partially labeled classification with Markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, pages 945–952, 2001.

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

W.S. Torgerson. Multidimensional scaling I: Theory and method. *Psychometrika*, 17:401–419, 1952.

L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. Dimensionality reduction: A comparative review. Online Preprint, 2008.

K.Q. Weinberger, F. Sha, and L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the* 21^{st} *International Confernence on Machine Learning*, 2004.

K.Q. Weinberger, F. Sha, Q. Zhu, and L.K. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems*, volume 19, 2007.

C.K.I. Williams. On a connection between Kernel PCA and metric multidimensional scaling. *Machine Learning*, 46(1-3):11–19, 2002.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the* 20^{th} *International Conference on Machine Learning*, pages 912–919, 2003.

Journal of Machine Learning Research 9 (2008) 2607-2633 Submitted 1/07; Revised 8/08; Published 11/08

Model Selection for Regression with Continuous Kernel Functions Using the Modulus of Continuity

Imhoi Koo IMHOI.KOO@KIOM.RE.KR
Department of Medical Research
Korea Institute of Oriental Medicine
Daejeon 305-811, Korea

Rhee Man Kil RMKIL@KAIST.AC.KR
Department of Mathematical Sciences
Korea Advanced Institute of Science and Technology
Daejeon 305-701, Korea

Editors: Isabelle Guyon and Amir Saffari

Abstract

This paper presents a new method of model selection for regression problems using the modulus of continuity. For this purpose, we suggest the prediction risk bounds of regression models using the modulus of continuity which can be interpreted as the complexity of functions. We also present the model selection criterion referred to as the modulus of continuity information criterion (MCIC) which is derived from the suggested prediction risk bounds. The suggested MCIC provides a risk estimate using the modulus of continuity for a trained regression model (or an estimation function) while other model selection criteria such as the AIC and BIC use structural information such as the number of training parameters. As a result, the suggested MCIC is able to discriminate the performances of trained regression models, even with the same structure of training models. To show the effectiveness of the proposed method, the simulation for function approximation using the multilayer perceptrons (MLPs) was conducted. Through the simulation for function approximation, it was demonstrated that the suggested MCIC provides a good selection tool for nonlinear regression models, even with the limited size of data.

Keywords: regression models, multilayer perceptrons, model selection, information criteria, modulus of continuity

1. Introduction

The task of learning from data is to minimize the expected risk (or generalization error) of a regression model (or an estimation function) under the constraint of the absence of *a priori* model of data generation and with the limited size of data. For this learning task, it is necessary to consider nonparametric regression models such as artificial neural networks, as the functional form of the target function is usually unknown. Furthermore, a mechanism to minimize the expected risk from the limited size of data is required. In this context, the model selection is an important issue in the selection of a reasonable network size in order to minimize the expected risk. However, the proper network size (or number of parameters) of a regression model is difficult to choose, as it is possible to obtain the empirical risk only in the case of the limited size of data while the expected risk of a regression model should be measured for the entire data distribution. For the expected risk of a regression model, the loss function of the error square is usually measured, and the expecta-

tion of the loss function for the entire data distribution is considered. This expected risk can be decomposed by the bias and variance terms of the regression models. If the number of parameters is increased, the bias term is decreased while the variance term is increased, and the opposite also applies. If the number of parameters is exceedingly small and the performance is thus not optimal due to a large bias term, a situation known as under-fitting of the regression models arises. If the number of parameters is especially large and the performance is thus not optimal due to a large variance term, over-fitting of the regression models arises. Hence, a trade-off exists between the under-fitting and over-fitting of regression models. Here, an important issue is measuring the model complexity associated with the variance term. Related to this issue, the statistical methods of model selection use a penalty term for the measurement of model complexity. Well known criteria using this penalty term are the Akaike information criterion (AIC) (Akaike, 1973), the Bayesian information criterion (BIC) (Schwartz, 1978), the generalized cross-validation (GCV) (Wahba et al., 1979), the minimum description length (MDL) (Rissanen, 1986; Barron et al., 1998), and the risk inflation criterion (RIC) (Foster and George, 1994). These methods can be well fitted with linear regression models when enough samples are available. However, they suffer the difficulty of selecting the optimal structure of the estimation networks in the case of nonlinear regression models and/or a small number of samples. For more general forms of regression models, Vapnik (1998) proposed a model selection method based on the structural risk minimization (SRM) principle. One of the characteristics of this method is that the model complexity is described by structural information such as the VC dimension of the hypothesis space associated with estimation networks, which indicates the number of samples that can be shattered, in other words, which can be completely classified by the given structure of estimation networks. This method can be applied to nonlinear models and also regression models trained for a small number of samples. For this problem, Chapelle et al. (2002), Cherkassky (1999), and Cherkassky and Ma (2003) showed that the SRM-based model selection is able to outperform other statistical methods such as AIC or BIC in regression problems with the limited size of data. On the other hand, these methods require the actual VC dimension of the hypothesis space associated with the estimation functions, which is usually not easy to determine in the case of nonlinear regression models. In this context, we consider the bounds on the expected risks using the modulus of continuity representing a measure of the continuity for the given function. Lorentz (1986) applied the modulus of continuity to function approximation theories. In the proposed method, this measure is applied to determine the bounds on the prediction risk. To be exact, it seeks the expected risk of an estimation function when predicting new observations. To describe these bounds, the modulus of continuity is analyzed for both the target and estimation functions, and the model selection criterion referred to as the modulus of continuity information criterion (MCIC) is derived from the prediction risk bounds in order to select the optimal structure of regression models. One of the characteristics in the suggested MCIC is that it can be estimated directly from the given samples and a trained estimation function. Through the simulation for function approximation using multi-layer perceptrons (MLPs), it is demonstrated that the suggested MCIC is effective for nonlinear model selection problems, even with the limited size of data.

This paper is organized as follows: in Section 2, we introduce the model selection criteria based on statistics such as the AIC and BIC, the model selection criteria based on Shannon's information theory such as the MDL, and the VC dimension based criteria. Section 3 describes the suggested model selection method referred to as the MCIC method starting from the definition of the modulus of continuity for continuous functions. We also describe how we can estimate the modulus of continuity for the regression models with different type of kernel functions. Section 4 describes

the simulation results for regression problems for various benchmark data using model selection methods including the suggested MCIC method. Finally, Section 5 presents the conclusion.

2. Model Selection Criteria for Regression Models

For the selection of regression models, the proper criteria for the decision methods are required. Here, various criteria used for the selection of regression models are described. First, let us consider a regression problem of estimating a continuous function f in $C(X,\mathbb{R})$ where $X \subset \mathbb{R}^m$ ($m \geqslant 1$) and $C(X,\mathbb{R})$ is a class of continuous functions. The observed output y for $\mathbf{x} \in X$ can be represented by

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon, \tag{1}$$

where $f(\mathbf{x})$ represents the target function and ε represents random noise with a mean of zero and a variance of σ_ε^2. Here, for regression problems, a data set $D = \{(\mathbf{x}_i, y_i) | \ i = 1, \cdots, N\}$, where $(\mathbf{x}_i, y_i)$ represents the ith pair of input and output samples, is considered. It is assumed that these pairs of input and output samples are randomly generated according to the distribution $P(\mathbf{x}),\ \mathbf{x} \in X$; that is,

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \ \ \mathbf{x}_i \in X, \tag{2}$$

where $\varepsilon_i,\ i = 1, \cdots, N$ represent independent and identically distributed (*i.i.d.*) random variables having the same distribution with ε. For these samples, our goal of learning is to construct an estimation function $f_n(\mathbf{x}) \in F_n$ (the function space with n parameters) that minimizes the expected risk

$$R(f_n) = \int_{X \times \mathbb{R}} L(y, f_n(\mathbf{x})) dP(\mathbf{x}, y) \tag{3}$$

with respect to the number of parameters n, where $L(y, f_n(\mathbf{x}))$ is a given loss functional, usually the square loss function $L(y, f_n(\mathbf{x})) = (y - f_n(\mathbf{x}))^2$ for regression problems. In general, an estimation function f_n can be constructed as a linear combination of kernel functions; that is,

$$f_n(\mathbf{x}) = \sum_{k=1}^{n} w_k \phi_k(\mathbf{x}), \tag{4}$$

where w_k and ϕ_k represent the kth weight value and kernel function, respectively.

To minimize the expected risk (3), it is necessary to identify the distribution $P(\mathbf{x}, y)$; however, this is usually unknown. Rather, we usually find f_n by minimizing the empirical risk $R_{emp}(f_n)$ evaluated by the mean of loss function values for the given samples; that is,

$$R_{emp}(f_n) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f_n(\mathbf{x}_i)). \tag{5}$$

Here, if the number of parameters n is increased, the empirical risk of (5) is decreased so that the bias of the estimation function is decreased while the variance of the estimation function is increased, or vice versa. Therefore, a reasonable trade-off must be made between the bias and variance in order to minimize the expected risk. One way of solving this problem is to estimate the expected risks for the given parameters of regression models. In statistical regression models, a popular criterion is the Akaike information criterion (AIC), in which an estimate of the expected risk is given by

$$\mathrm{AIC}(f_n) = R_{emp}(f_n) + 2 \cdot \frac{n}{N} \sigma_\varepsilon^2 \tag{6}$$

under the assumption that the noise term ε has a normal distribution. Here, the noise term $\hat{\sigma}_\varepsilon^2$ can be estimated by

$$\hat{\sigma}_\varepsilon^2 = \frac{RSS}{N - DoF(f_n)}, \tag{7}$$

where RSS represents the sum of the square error over the training samples; that is, $RSS = NR_{emp}(f_n)$, and $DoF(f_n)$ represents the degree of freedom of an estimation function f_n. This criterion is derived in the sense of the maximum-likelihood estimate of the regression parameters. As an alternative to this criterion, the Bayesian approach to model selection referred to as the Bayesian information criterion (BIC) can be considered:

$$\mathrm{BIC}(f_n) = R_{emp}(f_n) + \log N \cdot \frac{n}{N}\sigma_\varepsilon^2. \tag{8}$$

Compared to the AIC, the BIC treats complex models more heavily, giving preference to simpler models, when $N > e^2$, in which e represents the base of natural logarithms. Here, it is important to note that in both criteria, prior knowledge of the variance of noise term σ_ε^2 is needed or estimation of this term using (7) is required. These criteria are good for linear regression models with a large number of samples, as the AIC and BIC formulas hold asymptotically as the number of samples N goes to infinity.

As an alternative to the AIC or BIC, a frequently used model selection criterion is the minimum description length (MDL) criterion. In this method, for the regression model f_n and the data D, the description length $l(f_n, D)$ is described by

$$l(f_n, D) = l(D|f_n) + l(f_n),$$

where $l(f_n)$ represents the length of the regression model and $l(D|f_n)$ represents the length of the data given the regression model. According to Shannon's information theory, the description length in number of bits is then described by

$$l(f_n, D) = -\log_2 p(D|f_n) - \log_2 p(f_n),$$

where $p(f_n|D)$ represents the probability of the output data given the regression model and $p(f_n)$ represents a priori model probability. For a priori model probability, Hinton and Camp (1993) used a zero-mean Gaussian distribution for the neural network parameters. With the additional assumption that the errors of the regression model are $i.i.d.$ with a normal distribution, the description length of the regression model (Cohen and Intrator, 2004) can be described by

$$\mathrm{MDL}(f_n) = \log R_{emp}(f_n) + \frac{n}{N}\left(\log(2\pi) + \log(\frac{1}{n}\sum_{k=1}^{n} w_k^2) + 1\right). \tag{9}$$

This formula for the MDL shows that the description length of the regression model is composed of the empirical risk and the complexity term, which is mainly dependent upon the ratio of the number of parameters to the number of samples and the mean square of weight values. Here, minimizing the description length is equivalent to maximizing the posterior probability of the regression model. Hence the MDL method can be considered as another interpretation of the BIC method. In this context, a regression model that minimizes the description length should be chosen.

A good measure of model complexity in nonlinear models is the VC dimension (Vapnik, 1998) of the hypothesis space associated with estimation networks. The VC dimension can represent the

capacity (or complexity) of the estimation network in terms of the number of samples; that is, the maximum number of samples which can be shattered (classified in every possible way) by the estimation network. As the hypothesis space increases, the empirical risk can be decreased but the confidence interval associated with the complexity of the estimation network then increases. From this point of view, it is necessary to make a proper trade-off between the empirical risk and the confidence interval. The structural risk minimization (SRM) principle considers both the empirical risk and the complexity of the regression model to decide the optimal structure of the regression model. In this approach, for the VC dimension h_n measured for the hypothesis space F_n of regression models and the confidence parameter δ (a constant between 0 and 1), the expected risk satisfies the following inequality with a probability of at least $1-\delta$ (Vapnik, 1998; Cherkassky, 1999; Cherkassky and Ma, 2003):

$$R(f_n) \leqslant R_{emp}(f_n) \left(1 - c\sqrt{\frac{h_n(1+\ln(N/h_n)) - \ln\delta}{N}}\right)_+^{-1}, \tag{10}$$

where c represents a constant dependent on the norm and tails of the loss function distribution and $u_+ = \max\{u, 0\}$.

If the basis functions $\{\phi_1(\mathbf{x}), \cdots, \phi_n(\mathbf{x})\}$ are orthogonal with respect to the probability measure $P(\mathbf{x})$, the form of (10) can be described in a way that is easier to calculate. For the experimental set up, Chapelle et al. (2002) suggested the following bound with the confidence parameter $\delta = 0.1$:

$$R(f_n) \leqslant R_{emp}(f_n) T_{SEB}(n, N), \tag{11}$$

where

$$T_{SEB}(n,l) = \frac{1 + n/(NK)}{1 - (n/N)} \quad \text{and}$$

$$K = \left(1 - \sqrt{\frac{n(1+\ln(2N/n)) + 4}{N}}\right)_+.$$

These risk estimates of (10) and (11) were successfully applied to the model selection of regression problems with the limited size of data. In these risk estimates, the VC dimension of regression models should be estimated. For the case of nonlinear regression models such as artificial neural networks, the bounds on VC dimensions (Karpinski and Macintyre, 1995; Sakurai, 1995) can be determined. However, in general, it is difficult to estimate the VC dimension of nonlinear regression models accurately.

In this work, we consider a useful method for the selection of nonlinear regression models with the limited size of data. For this problem, the AIC or BIC method may not be effective in view of the fact that the number of samples may not be large enough to apply the AIC or BIC method. Moreover, an estimation of the VC dimension of nonlinear regression models is generally not straightforward. In this context, we consider to use the modulus of continuity representing a measure of continuity for the given function. In the proposed method, this measure is applied to determine the bounds on the prediction risk; that is, the expected risk of the estimation function when predicting new observations. From this result, a model selection criterion referred to as the modulus of continuity information criterion (MCIC) is suggested and it is applied to the selection of nonlinear regression models. The backgrounds and theories related to the suggested method are described in the next section.

3. Model Selection Criteria Based on the Modulus of Continuity

For the description of the bounds on expected risks, the modulus of continuity defined for continuous functions is used. In this section, starting from the definition of the modulus of continuity, the bounds on expected risks are described and the model selection criterion referred to as the MCIC using the described bounds is suggested.

3.1 The Modulus of Continuity for Continuous Functions

The modulus of continuity is a measure of continuity for continuous functions. First, it is assumed that X is a compact subset of Euclidean space $\mathbb{R}^m$; that is, the set X is bounded and closed in Euclidean space $\mathbb{R}^m$. Here, let us consider the case of univariate functions; that is, $m = 1$. Then, the measure of continuity $w(f,h)$ of a function $f \in C(X)$ can be described by the following form (Lorentz, 1986):

$$\omega(f,h) = \max_{x,x+t\in X,|t|\leqslant h} |f(x+t) - f(x)|, \tag{12}$$

where h is a positive constant. This modulus of continuity of f has the following properties:

- $\omega(f,h)$ is continuous at h for each f,
- $\omega(f,h)$ is positive and increases as h increases, and
- $\omega(f,h)$ is sub-additive; that is, $\omega(f,h_1+h_2) \leqslant \omega(f,h_1) + \omega(f,h_2)$ for positive constants h_1 and h_2.

As a function of f, the modulus of continuity has the following properties of a semi-norm:

$$\omega(af,h) \leqslant |a|\omega(f,h) \quad \text{for a constant } a \text{ and}$$

$$\omega(f_1+f_2,h) \leqslant \omega(f_1,h) + \omega(f_2,h) \quad \text{for } f_1 \text{ and } f_2 \in C(X).$$

One famous example of the modulus of continuity of a function f is that f is defined on $A = [a,b]$ ($b > a$) and satisfies a Lipschitz condition with the constant $M \geqslant 0$ and the exponent α ($0 < \alpha \leqslant 1$), denoted by $\mathrm{Lip}_M\alpha$; that is,

$$|f(a_1) - f(a_2)| \leqslant M|a_1 - a_2|^\alpha, \;\; a_1, a_2 \in A.$$

In this case, the modulus of continuity is given by

$$\omega(f,h) \leqslant Mh^\alpha.$$

In the multi-dimensional input spaces; that is, $X \subset \mathbb{R}^m$ ($m > 1$), there are different definitions of the modulus of continuity for a continuous function f. The following two definitions of the modulus of continuity are considered (Lorentz, 1986; Anastassiou and Gal, 2000):

Definition 1 *Let $m = 2$ and $X \subset \mathbb{R}^m$.*

- *Then, the modulus of continuity for $f \in C(X)$ is defined by*

$$\omega^A(f,h) = \sup\{|f(x_1,y_1) - f(x_2,y_2)|\}$$
$$\text{subject to } \begin{cases} (x_1,y_1),(x_2,y_2) \in X \text{ and} \\ \|(x_1,y_1) - (x_2,y_2)\|_2 \leqslant h, \;\; \text{for } h > 0. \end{cases}$$

- *Another definition of the modulus of continuity is*

$$\omega^B(f,\alpha,\beta) = \sup\left\{ \begin{array}{l} |f(x_1,y) - f(x_2,y)|, \\ |f(x,y_1) - f(x,y_2)| \end{array} \right\} \tag{13}$$
$$\text{subject to } \left\{ \begin{array}{l} (x_1,y),(x_2,y),(x,y_1),(x,y_2) \in X \text{ and} \\ |x_1 - x_2| \leqslant \alpha, |y_1 - y_2| \leqslant \beta, \text{for } \alpha,\beta > 0. \end{array} \right.$$

For $f \in C(X)$ on a compact subset $X \subset \mathbb{R}^m$, where $m > 2$, it is possible to define the modulus of continuity by induction.

The main difference in these two definitions of the modulus of continuity is the direction. The first definition measures the variation of all directions at some point $\mathbf{x} \in X$ while the second is dependent upon axis directions only at some point $\mathbf{x} \in X$. The relationship between the two definitions of the modulus of continuity can be described by the following lemma:

Lemma 1 *For $f \in C(X)$, two definitions of the modulus of continuity, $\omega^A(f,h)$ and $\omega^B(f,h,h)$ have the following relationship:*

$$\omega^B(f,h,h) \leqslant \omega^A(f,h) \leqslant 2\omega^B(f,h,h),$$

where h represents a positive constant.

For the proof of this lemma, refer to the Appendix A.1. Furthermore, each definition of the modulus of continuity has the following upper bound:

Lemma 2 *Let $f \in C^1(X)$, the class of continuous functions having continuous 1st derivative on X, a compact subset of $\mathbb{R}^m, m > 1$. Then, for $h > 0$, the modulus of continuity w^A and w^B have the following upper bounds:*

$$\omega^A(f,h) \leqslant h\sqrt{\sum_{i=1}^{m} \left\| \frac{\partial f}{\partial x_i} \right\|_\infty^2} \quad \textit{and}$$

$$\omega^B(f,h,\cdots,h) \leqslant h \max_{1 \leqslant i \leqslant m} \left\{ \left\| \frac{\partial f}{\partial x_i} \right\|_\infty \right\},$$

where x_i represents the ith coordinate in the point $\mathbf{x} = (x_1, x_2, \cdots, x_m) \in X$ and $\|\cdot\|_\infty$ represents the supremum norm (or L_∞ norm); that is, for a real- or complex-valued bounded function $g(\mathbf{x})$,

$$\|g\|_\infty = \sup\{|g(\mathbf{x})| \mid \mathbf{x} \in X_g\},$$

where X_g represents the domain of g.

For the proof of this lemma, refer to the Appendix A.2. From this lemma, the second definition of the modulus of continuity $w^B(f,h,h)$ was chosen because it has a smaller upper bound compared to the first modulus of continuity. For our convenience, the notation $w(f,h)$ is used to represent $w^B(f,h,\cdots,h)$ in the remaining sections of this paper.

The computation of the modulus of continuity requires the value of h. First, let us consider the following definition of a density of input samples (Tinman, 1963):

Definition 2 *The density D of an input sample set $\{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \subset X$, a compact subset of $\mathbb{R}^m, m > 1$, is defined by*

$$D(\mathbf{x}_1, \cdots, \mathbf{x}_N) = \sup_{\mathbf{x} \in X} \inf_{1 \leqslant i \leqslant N} d(\mathbf{x}_i, \mathbf{x}),$$

where $d(\mathbf{x}_i, \mathbf{x})$ represents the distance between $\mathbf{x}_i$ and $\mathbf{x} \in X$, which is explicitly any metric function such that, for every $\mathbf{x}, \mathbf{y}, \mathbf{z} \in X$, the following properties are satisfied: $d(\mathbf{x}, \mathbf{y}) \geqslant 0$ with the equality if and only if $\mathbf{x} = \mathbf{y}$, $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$, and $d(\mathbf{x}, \mathbf{z}) \leqslant d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$.

Let us also consider a point $\mathbf{x}_0 \in X$ such that

$$\mathbf{x}_0 = \arg\max_{\mathbf{x} \in X} |f(\mathbf{x}) - f_n(\mathbf{x})|.$$

Then, the value of h can be bounded by

$$\min_{1 \leqslant i \leqslant N} d(\mathbf{x}_i, \mathbf{x}_0) \leqslant h \leqslant D(\mathbf{x}_1, \cdots, \mathbf{x}_N) \tag{14}$$

to cover the input space X using balls $B(\mathbf{x}_i, h)$ with centers as input samples $\mathbf{x}_i$ and a radius of h: $B(\mathbf{x}_i, h) = \{\mathbf{x} \,|\, \|\mathbf{x}_i - \mathbf{x}\| < h\}$. This range of h is considered to describe the modulus of continuity for the target and estimation functions.

3.2 Risk Bounds Based on the Modulus of Continuity

In this subsection, the modulus of continuity for the target and estimation functions are investigated, and the manner in which they are related to the expected risks is considered. First, let us consider the loss function for the observed model y and the estimation function $f_n(\mathbf{x})$ with n parameters as $L(y, f_n) = |y - f_n(\mathbf{x})|$. Then, the expected and the empirical risks are defined by the following L_1 measure:

$$R(f_n)_{L_1} = \int_{X \times \mathbb{R}} |y - f_n(\mathbf{x})| dP(\mathbf{x}, y) \quad \text{and}$$

$$R_{emp}(f_n)_{L_1} = \frac{1}{N} \sum_{i=1}^{N} |y_i - f_n(\mathbf{x}_i)|.$$

In the first step, let us consider the case of a univariate target function; that is, $f \in C(X)$ with $X \subset \mathbb{R}$. Then, with the definition of the modulus of continuity of (12) and the bound of h as described by (14), the relationship between the expected and empirical risks is described using the modulus of continuity as follows:

Theorem 1 *Let the target function $f \in C^1(X)$ of (1) with X, a compact subset of $\mathbb{R}$, be approximated by the estimation function f_n of (4), that is, a linear combination of weight parameters w_k and basis functions ϕ_k, $k = 1, \cdots, n$ for the given samples (x_i, y_i), $i = 1, \cdots, N$ generated by (2). Then, for the confidence parameters δ (a constant between 0 and 1), the expected risk in the L_1 sense is bounded by the following inequality with a probability of at least $1 - 2\delta$:*

$$R(f_n)_{L_1} \leqslant R_{emp}(f_n)_{L_1} + \frac{1}{N^2} \sum_{i,j=1}^{N} (|y_i - y_j| + |f_n(x_i) - f_n(x_j)|)$$

$$+ (\omega(f_n, h_0) + C) \sqrt{\frac{1}{2N} \ln \frac{2}{\delta}} \quad \textit{and} \tag{15}$$

$$C = |f_n(x_0) - f_n(x_0')| + 2\|f\|_\infty + 2\sigma_\varepsilon \sqrt{\frac{1}{\delta}} \quad \textit{for } x_0, x_0' \in \{x_1, \cdots, x_N\},$$

where $w(f_n, h_0)$ represents the modulus of continuity of the estimation function f_n and h_0 represents a constant satisfying (14).

For the proof of this theorem, refer to the Appendix A.3. This theorem states that the expected risk $R(f_n)_{L_1}$ is bounded by the empirical risk $R_{emp}(f_n)_{L_1}$, the second term of (15) representing the variations of output samples and also the variations of estimation function values for the given input samples, and the third term representing the modulus of continuity for the estimation function $w(f_n, h_0)$ and a constant C associated with target function. Here, let us consider the second term. By investigating this term further, it can be decomposed it into the empirical risk and the term depending on the target function. The next corollary shows the bounds on the expected risks with this decomposition:

Corollary 1 *Let H_y be the $N \times N$ matrix in which the ij-th entry is given by $|y_i - y_j|$. Then, the following inequality holds with a probability of at least $1-2\delta$:*

$$R(f_n)_{L_1} \leqslant 3R_{emp}(f_n)_{L_1} + \frac{2}{N}\max\{\lambda_i\} + (\omega(f_n, h_0) + C)\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}, \tag{16}$$

where λ_i represents the ith eigenvalue of the matrix H_y.

For the proof of this lemma, refer to the Appendix A.4. This corollary states that the dominant terms related to the estimation function f_n in the expected risk are the empirical risk $R_{emp}(f_n)$ and the modulus of continuity $w(f_n, h_0)$, as the eigenvalue λ_i of H_f is not dependent upon f_n and a constant C has little influence on the shape of expected risks as the number of parameters n increases. The bounds on the expected risks of (16) appear to be overestimated, as the empirical risk is multiplied by 3. However, for the purpose of determining the model selection criteria, an estimation of the tight bound on the expected risk is not essential. Rather, the coefficient ratio between the empirical risk and modulus of continuity terms plays an important role for model selection problems because only these two terms are mainly dependent upon the estimation function f_n. From this point of view, the following model selection criterion referred to as the modulus of continuity information criterion (MCIC) is suggested:

$$\mathrm{MCIC}(f_n) = R_{emp}(f_n)_{L_1} + \frac{\omega(f_n, h_0)}{3}\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}. \tag{17}$$

Suppose we have fixed number of samples N. Then, as the number of parameters n increases, the empirical risk $R_{emp}(f_n)$ decreases while the modulus of continuity $\omega(f_n, h_0)$ increases, as the estimation function f_n becomes a more complex function. Accordingly, it is necessary to make a trade-off between the over-fitting and under-fitting of regression models using the MCIC for the optimization of the regression models.

This far, univariate continuous functions are addressed. At this point, let us consider the case of $X \subset \mathbb{R}^m$ with $m > 1$; that is, the case of multivariate continuous functions. Here, it is possible to show that the prediction risk bounds take a similar form to those of univariate continuous functions. The following theorem of the prediction risk bounds for multivariate continuous functions is suggested using the definition of the modulus of continuity (13):

Theorem 2 *Let $f \in C^1(X)$ with X, a compact subset of $\mathbb{R}^m$ ($m > 1$), and h_0 be a constant satisfying (14). Then, for the confidence parameter δ (a constant between 0 and 1), the expected risk in L_1 sense is bounded by the following inequality with a probability of at least $1-\delta$:*

$$R(f_n)_{L_1} \leqslant R_{emp}(f_n)_{L_1} + \left\{\omega(f - f_n, h_0) + |f(\mathbf{x}_{i_0}) - f_n(\mathbf{x}_{i_0})| + \sigma_\varepsilon\sqrt{\frac{2}{\delta}}\right\}\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}},$$

where $w(f-f_n,h_0)$ represents the modulus of continuity of the function $f-f_n$, h_0 represents a constant satisfying (14), and $\mathbf{x}_{i_0}$ represents an element of an input sample set $\{\mathbf{x}_1,\cdots,\mathbf{x}_N\}$.

For the proof of this theorem, refer to the Appendix B.1. In this theorem, $w(f-f_n,h_0)$ can be replaced with $w(f,h_0)+w(f_n,h_0)$; that is, the sum of the modulus of continuity for the target and estimation functions because the following inequalities always hold:

$$\omega(f_n,h_0)-\omega(f,h_0)\leqslant\omega(f-f_n,h_0)\leqslant\omega(f_n,h_0)+\omega(f,h_0).$$

The suggested theorem states that the expected risk is mainly bounded by the empirical risk $R_{emp}(f_n)$, the modulus of continuity for the target function $w(f,h_0)$, and also the modulus of continuity for the estimation function $w(f_n,h_0)$. As the number of parameters n varies, the empirical risk, the modulus of continuity for the estimation function, and the term $|f(\mathbf{x}_{i_0})-f_n(\mathbf{x}_{i_0})|$ are changed while other terms remain constant. Here, in order to find the optimal model complexity $n=n^*$ that minimizes expected risk $R(f_n)$, these varying terms should be considered. In this case, the effect of the term $|f(\mathbf{x}_{i_0})-f_n(\mathbf{x}_{i_0})|$ is small compared with the other two terms, as the regression model becomes well fitted to the samples as the number of parameters n increases. This implies that the model selection criteria for multivariate estimation functions have the same form as (17) except with a coefficient $1/3$ of $\omega(f_n,h_0)$. In practice, the performance of MCIC for model selection problems is not so sensitive to this coefficient. Summarizing the properties of the suggested MCIC, the distinct characteristics are described as follows:

- The suggested MCIC is dependent upon the modulus of continuity for the trained estimation function.
- The suggested MCIC is also dependent upon the value of h_0 which varies according to the sample distribution.

Considering these characteristics, for model selection problems, the MCIC is a measure sensitive to the trained estimation function using a certain learning algorithm and also sensitive to the distribution of samples while other model selection criteria such as the AIC and BIC depend on structural information such as the number of parameters. For the computation of the suggested MCIC, the modulus of continuity of the trained estimation function should be evaluated, as explained in the next subsection.

3.3 The Modulus of Continuity for Estimation Functions

The modulus of continuity for the estimation function $w(f_n,h)$ is dependent upon the basis function ϕ_k in (4). Here, examples of computing $w(f_n,h)$ according to the type of basis functions are presented:

- A case of the estimation function f_n with algebraic polynomials on $X=[a,b]\subset\mathbb{R}$:

$$\phi_k(x)=x^k \quad \text{for } k=0,1,\cdots,n.$$

  Applying the mean value theorem to ϕ_k, we get

$$\begin{aligned}\omega(\phi_k,h)&\leqslant\left\|\phi_k'\right\|_\infty\cdot h\\&\leqslant kh\cdot\max\left\{|a|^{k-1},|b|^{k-1}\right\},\ k=1,\cdots,n.\end{aligned}$$

Therefore, the modulus of continuity for f_n has the following upper bound:

$$\omega(f_n,h) \leqslant \sum_{k=1}^{n} kh|w_k| \cdot \max\left\{|a|^{k-1},|b|^{k-1}\right\}.$$

- A case of the estimation function f_n with trigonometric polynomials $\phi_k(x)$ on $X \subset \mathbb{R}$:

$$\phi_k(x) = \begin{cases} 1/2 & \text{if } k=0 \\ \sin\left((k+1)x/2\right) & \text{if } k = \text{odd number} \\ \cos\left(kx/2\right) & \text{if } k = \text{even number} \end{cases}$$

for $k = 1,\cdots,n$. Applying the mean value theorem to ϕ_k, we get

$$\begin{aligned} \omega(\phi_k,h) &\leqslant \|\phi_k'\|_\infty h \\ &\leqslant \left\lfloor \frac{k}{2} \right\rfloor h, \text{ for } k=1,\cdots,n. \end{aligned}$$

Therefore, the modulus of continuity for f_n has the following upper bound:

$$\omega(f_n,h) \leqslant \sum_{k=0}^{n} h|w_k| \cdot \left\lfloor \frac{k}{2} \right\rfloor.$$

- A case of the estimation function f_n with sigmoid function $\phi_k(\mathbf{x}) = \phi_k(x_1,\cdots,x_m)$ on $X \subset \mathbb{R}^m$:

$$f_n(\mathbf{x}) = \sum_{k=1}^{n} w_k \phi_k(x_1,\cdots,x_m) + w_0,$$

where

$$\phi_k(x_1,\cdots,x_m) = \tanh\left(\sum_{j=1}^{m} v_{kj}x_j + v_{k0}\right).$$

Applying the mean value theorem to ϕ_k with respect to each coordinate $x_1,\cdots,x_m$, we get

$$|f_n(\cdots,x_j,\cdots) - f_n(\cdots,x_j-h,\cdots)| \leqslant h \cdot \left\| \frac{\partial f_n}{\partial x_j} \right\|_\infty \text{ for } j=1,\cdots,m.$$

Therefore, the modulus of continuity for f_n has the following upper bound:

$$\begin{aligned} \omega(f_n,h) &\leqslant h \cdot \max_{1\leqslant j\leqslant m} \left\| \frac{\partial f_n}{\partial x_j} \right\|_\infty \\ &\leqslant h \cdot \max_{1\leqslant j\leqslant m} \left\| \sum_{k=1}^{n} w_k v_{kj} \cdot \left(1 - \tanh^2\left(\sum_{i=1}^{m} v_{ki}x_i + v_{k0}\right)\right) \right\|_\infty \\ &\leqslant h \cdot \max_{1\leqslant j\leqslant m} \sum_{k=1}^{n} \left|w_k v_{kj}\right|. \end{aligned} \tag{18}$$

As shown in these examples, the modulus of continuity for the estimation function f_n is dependent upon the trained parameter values associated with f_n and h_0 whose range is given by (14). However, the proper value of h_0 satisfying (14) requires the intensive search of the input space. From this point of view, in practice, the value of h_0 is considered as the half of the average distance between two adjacent samples. Assuming a uniform distribution of input samples, the value of h_0 can be determined from the range of data values in each coordinate. For example, for m dimensional input patterns, h_0 can be determined by

$$h_0 = \frac{1}{2}\left(\frac{1}{m}\sum_{i=1}^{m}\frac{\max_i - \min_i}{N-1}\right)^{1/m}, \tag{19}$$

where $\max_i$ and $\min_i$ represent the maximum and minimum values of the samples in the ith coordinate.

After the value of h_0 is determined, the computation of the modulus of continuity requires access to all the parameter values of f_n which are obtained after the learning of training samples. In this context, the computational complexity of the modulus of continuity is proportional to the number of parameters n in the estimation function, that is, in big-O notation, $O(n)$. This computational complexity is not so heavy compared to the calculation of the empirical risk term, as it requires the computational complexity of $O(N)$ and in general, $N \gg n$. Hence, in total, the computational complexity of MCIC is described by $O(N)$ which is equivalent to the computational complexity of the AIC or BIC.

Once the modulus of continuity is determined, the MCIC can be determined by (17). Then, the model with the smallest value of MCIC will then be selected. Here, $\widehat{n}$ is selected such that

$$\widehat{n} = \arg\min_n \mathrm{MCIC}(f_n).$$

The validity of the suggested MCIC is shown in the next section through the simulation for nonlinear model selection problems.

4. Simulation

The simulation for function approximation was performed using the multilayer perceptrons (MLPs) composed of the input, hidden, and output layers. For this simulation, the number of sigmoid units n in the hidden layer was increased from 1 to 50. Here, f_n was denoted as the MLP with a hidden layer including n sigmoid units with the m dimensional input. The functional form of the estimation function is given by

$$f_n(x) = \sum_{k=1}^{n} w_k \tanh\left(\sum_{j=1}^{m} v_{kj}x_j + v_{k0}\right) + w_0,$$

where v_{kj} and w_k represent the input and output weights, respectively, that are associated with the kth sigmoid unit, and v_{k0} and w_0 represent the bias terms of the kth sigmoid unit and of the estimation function, respectively. In this regression model, the conjugate gradient method was used for the training of the input and output weights of the MLPs. As for the different type of kernel functions, we presented the model selection method using the suggested MCIC for the regression model with trigonometric polynomials (Koo and Kil, 2006) and showed the effectiveness of the MCIC compared to other model selection criteria.

As the benchmark data for this simulation of function approximation, the target functions given by Donoho and Johnstone (1995) were used: they are Blocks, Bumps, Heavysine, and Doppler functions, as illustrated in Figure 1. To generate the data for each target function from D-J (Donoho and Johnstone), the input values x_i, $i = 1, \cdots, N$ were generated from a uniform distribution within the interval of $[0, 2\pi]$. Here, for the normalization of D-J data, the outputs were adjusted to have the mean square value of 1 within the interval of $[0, 2\pi]$. The noise terms were also generated from a normal distribution with a mean of zero and a standard deviation of $\sigma_\varepsilon = 0.2$ or 0.4. They were then added to the target values computed from the randomly generated input values. For these target functions, 100 sets of N (= 200) training samples were generated randomly to train the MLP. In addition to the training samples, 1000 test samples were also generated separately according to identical input and noise distributions.

As for another application of the MCIC, simulations for the target functions with binary output values were considered using the benchmark data suggested by Hastie et al. (2003): the target value is defined by

$$y(\mathbf{x}) = \begin{cases} 1 & \text{if} \sum_{j=1}^{10} x_j > 5 \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathbf{x}$ is uniformly generated in $[0,1]^{20}$. For the training of this target function, 100 sets of 50 samples were generated. In addition to the training samples, 500 test samples were also generated separately. The noise terms were also generated from a normal distribution with a mean of 0 and a standard deviation of $\sigma_\varepsilon = 0.0$ or 0.2. They were then added to the target values computed from the randomly generated input values.

For the simulation of selecting regression models with multi-dimensional input data, the benchmark data suggested by Chang and Lin (2001) were also used: they are Abalone, CPU_Small, MG, and Space_GA data sets as described in Table 1. For each data set, 30 sets of 500 samples were randomly generated as the training samples and the remaining samples in each set were used as the test samples; that is, 30 sets of test samples were also generated. For these data sets, the range of input and output was normalized between -1 and 1.

Data Set	Description	No. of Features	No. of Data
Abalone	predicting the age of abalone	8	4177
CPU_Small	predicting a computer system activity	12	8192
MG	predicting the Mackey-Glass time series	6	1385
Space_GA	election data on 3107 US counties	6	3107

Table 1: The benchmark data sets for regression problems

For our experiments, various model selection methods such as the AIC, BIC, MDL, and the suggested MCIC-based methods were tested. Once the MLP was trained, the empirical risk $R_{emp}(f_n)$ evaluated by the training samples was obtained, and the estimated risk $\widehat{R}(f_n)$ value could then be determined by the AIC, BIC, MDL, and MCIC-based methods. In the cases of the AIC and BIC methods, we selected the estimation function $f_{\hat{n}}$ which gives the smallest value of information criterion described as (6) and (8), respectively. In these criteria, we assume that the noise variance σ_ε^2 value was known. In the case of MDL, we selected the estimation function $f_{\hat{n}}$ which gives the smallest value of the description length of (9). In the suggested method, we used the following form

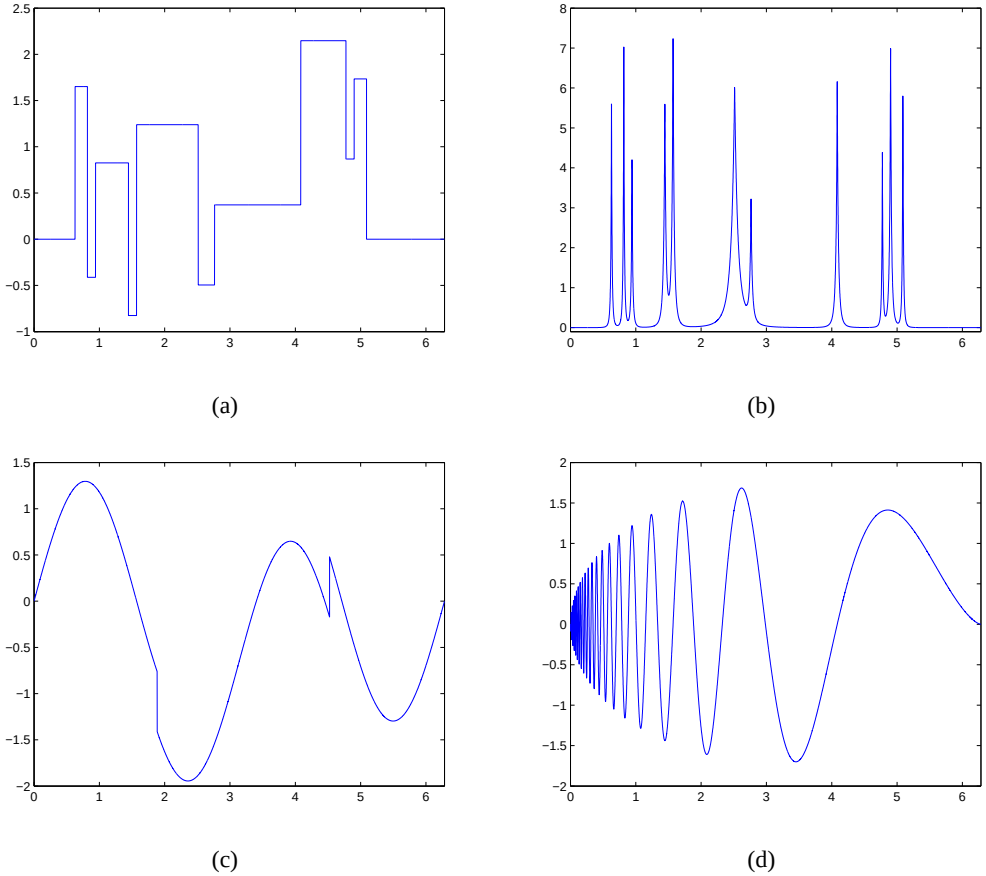


Figure 1: Target functions from Donoho and Johnstone (1995): (a), (b), (c), and (d) represent the Blocks, Bumps, Heavysine, and Doppler functions respectively.

of MCIC for MLPs using the modulus of continuity described as (18):

$$\mathrm{MCIC}(f_n) = R_{emp}(f_n)_{L_1} + \frac{h_0}{3} \max_{1 \leqslant j \leqslant m} \sum_{k=1}^{n} \left| w_k v_{kj} \right| \sqrt{\frac{1}{2N} \ln \frac{2}{\delta}}, \tag{20}$$

where h_0 was set to the half of the average distance between two adjacent samples using (19) and δ was set to 0.05. In our case, we selected $f_{\hat{n}}$ which gave the smallest value of (20).

To compare the performance of the model selection methods, the risks for the selected $f_{\hat{n}}$ were evaluated by the test samples and the results were compared with the minimum risk among all risks for f_n, $n = 1, \cdots, 50$. Quantitatively, the log ratio r_R of two risks $R_{test}(f_n)$ and $\min_n R(f_n)$ were computed:

$$r_R = \log \frac{R_{test}(f_{\hat{n}})}{\min_n R_{test}(f_n)}, \tag{21}$$

where R_{test} represents the empirical risk for the squared error loss function evaluated by the test samples. This risk ratio represents the quality of the estimated distance between the optimal and the estimated optimal risks.

Target Functions	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
Blocks	**0.0255**	0.0262	0.0416	0.0411	0.1459	0.0763	0.0371	0.0395
Bumps	**0.0393**	0.0697	0.0507	0.0718	0.1646	0.1294	0.0458	0.0426
Heavysine	0.0420	0.0537	0.1059	0.1147	0.1071	0.1225	**0.0107**	0.0160
Doppler	0.0343	0.0523	0.0932	0.0973	0.2318	0.1345	**0.0222**	0.0359

Table 2: Risk ratios for the regression of the four D-J target functions with $\sigma_\varepsilon = 0.2$.

Target Functions	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
Blocks	0.0441	0.0350	0.0853	0.0509	0.1146	0.0563	**0.0262**	0.0302
Bumps	**0.0511**	0.0572	0.0804	0.0699	0.1451	0.0810	0.0516	0.0433
Heavysine	0.0846	0.0616	0.1483	0.0773	0.1458	0.0762	**0.0130**	0.0151
Doppler	0.0801	0.0728	0.1421	0.0860	0.2225	0.1164	**0.0218**	0.0311

Table 3: Risk ratios for the regression of the four D-J target functions with $\sigma_\varepsilon = 0.4$.

After all experiments had been repeated for the given number of training sample sets, the means and standard deviations of the risk ratios of (21) for each target function were presented. First, in the case of D-J data sets, the simulation results of the model selection using the AIC, BIC, MDL, and MCIC based methods are presented in Tables 2 and 3. These simulation results showed that the suggested MCIC method provided the top level performances in all cases except the blocks and bumps target functions when $\sigma_\varepsilon = 0.2$ in which the AIC method showed the best performances. This was mainly due to the fact that the known noise standard deviation of σ_ε was used in the AIC method. To clarify this fact, another simulation for these target functions in which the AIC and BIC methods with the estimation of noise variances using (7) were used. These simulation results are presented in Table 4. In this simulation, as we expected, the suggested MCIC method showed the best performance.

We also observed the dependency of the number of samples during the selection of regression models. For this simulation, the numbers of samples were set to $N = 100$, 200, 400, and 800 for the regression of the Doppler target function with a noise standard deviation of $\sigma_\varepsilon = 0.4$. The simulation results are presented in Table 5. Here, note that the complexity of the doppler target function increases as the input value decreases. These results showed that the performances of the AIC, BIC, MDL, and MCIC methods were improved as the number of samples becomes larger as shown in Table 5. Among these model selection methods, the MCIC method always showed the better performances compared to other model selection methods, even in the smaller numbers of samples. This is mainly due to the fact that in the MCIC method, the modulus of continuity, which can be interpreted as the complexity of the estimation function was computed for each trained estimation function directly.

In the case of Hastie et al.'s benchmark data, the MDL model selection methods showed some merits in performances compared to other model selection methods as shown in Table 6. One of the reasons why the MCIC method does not show the better performances in this target function compared to other model selection methods is that this target function can be properly solved by the classification problem (not regression problem) in which the discriminant function is linear.

Target Functions	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
Blocks	0.0718	0.0612	0.1117	0.0744	0.1459	0.0763	**0.0371**	0.0395
Bumps	0.1107	0.0933	0.1558	0.1395	0.1646	0.1294	**0.0458**	0.0426

Table 4: Risk ratios for the regression of blocks and bumps functions with $\sigma_\varepsilon = 0.2$ using the AIC and BIC methods with the estimation of noise variances using (7), and the MDL and MCIC methods.

N	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
100	0.1072	0.0820	0.1552	0.1119	0.3205	0.1710	**0.0794**	0.0696
200	0.0801	0.0728	0.1421	0.0860	0.2225	0.1164	**0.0218**	0.0311
400	0.0477	0.0478	0.1028	0.0670	0.1588	0.0759	**0.0077**	0.0135
800	0.0174	0.0275	0.0610	0.0565	0.0860	0.0774	**0.0035**	0.0107

Table 5: The variation of risk ratios for the regression of Doppler function with $\sigma_\varepsilon = 0.4$ using the AIC, BIC, MDL, and MCIC methods according to the number of samples $N = 100$, 200, 400, and 800.

However, even in this case, if the sample size is reduced, the proposed method can have the merits in performances since the MCIC includes the complexity term of the estimation function using the modulus of continuity and for smaller number of samples, this complexity term has the high influence on the bounds on the expected risk. To clarify this fact, we made another simulation results for the number of samples reduced by half; that is, $N = 25$ and compared with the previous simulation results as shown in Table 7. These simulation results showed that the MCIC method demonstrated the better performances compared to other model selection methods by reducing the sample size.

σ_ε	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
0.0	0.3161	0.2197	0.3161	0.2197	**0.3100**	0.2273	0.4307	0.2624
0.2	0.3400	0.5028	0.3115	0.4658	**0.1881**	0.1175	0.3034	0.1503

Table 6: Risk ratios for the regression of the binary target function using the AIC, BIC, MDL, and MCIC methods when the number of samples N is 50.

The simulation results for the selection of regression models with multi-dimensional input data are summarized in Table 8. These simulation results showed that the suggested MCIC method achieved top or second level performances compared to other model selection methods. As shown in the previous case; that is, the regression problem of Hastie et al.'s benchmark data, the MCIC method is more effective when the sample size is small. To see the effect on smaller number of samples, we also made another simulation results for the number of samples reduced by half; that is,

	AIC		BIC		MDL		MCIC	
σ_ε	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
0.0	0.3620	0.1977	0.3620	0.1977	0.3430	0.1855	**0.2652**	0.1589
0.2	1.7428	0.9582	1.7428	0.9582	0.3169	0.1816	**0.2716**	0.1743

Table 7: The variation of risk ratios for the regression of the binary target function when the number of samples is reduced by half; that is, $N = 25$.

$N = 250$ and compared with the previous simulation results as shown in Table 9. These simulation results showed that the MCIC method demonstrated the top level performances compared to other model selection methods. All of these observations support that the suggested MCIC method is quite effective for nonlinear regression models especially for smaller number of samples. This is mainly due to the fact that the complexity term as a form of the modulus of continuity of the trained regression model provides high influence on selecting the regression model especially for smaller number of samples. This can be explained by the following observations:

- Once the estimation function is trained, the estimation function provides accurate values for the training samples. In this estimation function, the variation of the predicted values for the unobserved data with respect to the function values for the known data (or training samples) can be described by the modulus of continuity, as presented in the definition of the modulus of continuity.
- If the number of samples decreases, the density of input space becomes low and it makes a big value of h. Then, in the suggested MCIC, this makes high influence of the modulus of continuity compared to the empirical risk which usually has a small value.
- If there are enough number of samples for the target function, the opposite phenomenon of the above case happens.

In summary, through the simulation for function approximation using the MLPs, we have shown that the suggested MCIC provides performance advantages for the selection of regression models compared to other model selection methods in various situations of benchmark data. Compared to other model selection methods, the MCIC methods provides the considerable merits in performances especially when no knowledge of noise variances for the given samples is available and also when not enough number of samples considering the complexity of target function is available.

5. Conclusion

We have suggested a new method of model selection in regression problems based on the modulus of continuity. The prediction risk bounds are investigated from a view point of the modulus of continuity for the target and estimation functions. We also present the model selection criterion referred to as the MCIC which is derived from the suggested prediction risk bounds. The suggested MCIC is sensitive to the trained regression model (or estimation function) obtained from a specific learning algorithm and is also sensitive to the distribution of samples. As a result, the suggested MCIC is able to discriminate the performances of the trained regression models, even with the same structure of regression models. To verify the validity of the suggested criterion, the selection

Data Set	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
Abalone	0.0428	0.0586	0.0496	0.0410	0.0477	0.0493	**0.0324**	0.0303
CPU_Small	0.1646	0.1578	0.1212	0.1158	**0.0940**	0.0941	0.0941	0.1076
MG	0.0665	0.0442	0.0649	0.0371	0.0523	0.0470	**0.0449**	0.0343
Space_GA	**0.0851**	0.0612	0.1597	0.0869	0.1039	0.0626	0.0870	0.0526

Table 8: The variation of risk ratios for the regression of the benchmark data sets using the AIC, BIC, MDL, and MCIC methods when the number of samples N is 500.

Data Set	AIC		BIC		MDL		MCIC	
	mean	s. dev.	mean	s. dev.	mean	s. dev.	mean	s. dev.
Abalone	0.1385	0.1947	0.0668	0.1019	0.0618	0.0966	**0.0307**	0.0418
CPU_Small	0.2832	0.2906	0.2864	0.2646	0.2828	0.2929	**0.1942**	0.2219
MG	0.1451	0.1037	0.0816	0.0947	0.0887	0.0934	**0.0456**	0.0508
Space_GA	0.0768	0.0618	0.1466	0.0872	0.0801	0.0651	**0.0659**	0.0518

Table 9: The variation of risk ratios for the regression of the benchmark data sets when the number of samples is reduced by half; that is, $N = 250$.

of regression models using the MLPs that were applied to function approximation problems was performed. Through the simulation for function approximation using the MLPs, it was shown that the model selection method using the suggested MCIC has the advantages of risk ratio performances over other model selection methods such as the AIC, BIC, and MDL methods in various situations of benchmark data. Compared to other model selection methods, this merit of regression performances is significant especially when not enough number of samples considering the complexity of target function is available. Furthermore, the suggested MCIC method does not require any knowledge of a noise variance of samples which is usually given or estimated in other model selection methods. For regression models with other types of estimation functions that have some smoothness constraints, the suggested MCIC method can be easily extended to the given regression models by evaluating the modulus of continuity for the corresponding estimation functions.

Acknowledgments

The authors would like to thank to the editors and the anonymous reviewers for their helpful comments. This work was partially supported by the Korea Science and Engineering Foundation (KOSEF) grant (M10643020004-07N4302-00400) funded by the Korean Ministry of Education, Science, and Technology.

Appendix A.

In this appendix, we prove the lemmas 1 and 2 in Section 3.1. We also prove the theorem 1 and corollary 1 in Section 3.2; that is, the case of univariate target functions.

A.1 Proof of Lemma 1

Since $\omega^A(f,h)$ are considered all directions on h-ball on X, the following inequality always holds:

$$\omega^B(f,h,h) \leqslant \omega^A(f,h).$$

From the triangular inequality, the following inequality holds:

$$|f(x_1,y_1) - f(x_2,y_2)| \leqslant |f(x_1,y_1) - f(x_1,y_2)| + |f(x_1,y_2) - f(x_2,y_2)|.$$

Let $\|(x_1,y_1) - (x_2,y_2)\| \leqslant h$. Then, $|x_1 - x_2| \leqslant h$ and $|y_1 - y_2| \leqslant h$. Therefore, from the definition of the modulus of continuity, we obtain

$$\omega^A(f,h) \leqslant 2\omega^B(f,h,h).$$

□

A.2 Proof of Lemma 2

- Upper bound of $w^A(f,h)$: Let $\mathbf{x} \in X$ and $\mathbf{x} - \mathbf{h} \in X$ satisfying $\|\mathbf{h}\| \leqslant h$. Then,

$$\begin{aligned}
|f(\mathbf{x}) - f(\mathbf{x}-\mathbf{h})| &\leqslant |\nabla f(\mathbf{x} - \xi\mathbf{h}) \cdot \mathbf{h}| \quad \text{for some } \xi \in (0,1) \\
&\leqslant \|\mathbf{h}\| \, \|\nabla f(\mathbf{x} - \xi\mathbf{h})\| \\
&\qquad \text{(because of Cauchy-Schwartz inequality)} \\
&= \|\mathbf{h}\| \sqrt{\sum_{i=1}^{m} \left| \frac{\partial f}{\partial x_i}(\mathbf{x} - \xi\mathbf{h}) \right|^2} \\
&\leqslant \|\mathbf{h}\| \sqrt{\sum_{i=1}^{m} \left\| \frac{\partial f}{\partial x_i} \right\|_\infty^2}.
\end{aligned}$$

  Since the last term of the above equation is independent of $\mathbf{x} \in X$, we can conclude that

$$\omega^A(f,h) \leqslant h \sqrt{\sum_{i=1}^{m} \left\| \frac{\partial f}{\partial x_i} \right\|_\infty^2}.$$

- Upper bound of $w^B(f,h,h)$: Let $\alpha \in \mathbb{R}$ and $|\alpha| \leqslant h$. Here, let us define $\mathbf{e}_i$ as a vector on $\mathbb{R}^m$ whose i-th coordinate is 1 and the others are 0. Then, there exists $\xi \in (0,1)$ such that

$$|f(\mathbf{x}) - f(\mathbf{x} - h\mathbf{e}_i)| \leqslant \left| \frac{\partial f}{\partial x_i}(\mathbf{x} - \xi\alpha\mathbf{e}_i) \right| |h| \quad \text{for } i = 1, \cdots, m.$$

  This implies that

$$\max_i \{|f(\mathbf{x}) - f(\mathbf{x} - \alpha\mathbf{e}_i)|\} \leqslant |\alpha| \max_{1 \leqslant i \leqslant m} \left\{ \left| \frac{\partial f}{\partial x_i}(\mathbf{x} - \xi\alpha\mathbf{e}_i) \right| \right\}.$$

  Therefore, we can conclude that

$$\omega^B(f,h,\cdots,h) \leqslant h \max_{1 \leqslant i \leqslant m} \left\{ \left\| \frac{\partial f}{\partial x_i} \right\|_\infty \right\}.$$

□

A.3 Proof of Theorem 1

Before the description of main proof, let us introduce the Hoeffding inequality (Hoeffding, 1963): Given *i.i.d.* random variables $Y_1, \ldots, Y_N$, let us define a new random variable

$$S_N = \frac{1}{N}\sum_{i=1}^{N} Y_i$$

and we assume that there exist real numbers a_i and b_i for $i = 1, \ldots, N$ such that $\Pr\{Y_i \in [a_i, b_i]\} = 1$. Then, for any $\varepsilon > 0$, we have

$$\Pr\{E[S_N] - S_N \geqslant \varepsilon\} \leqslant \exp\left(-\frac{2\varepsilon^2 N^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right).$$

First, let us consider the noiseless case; that is, $y = f(x)$ in (1). For the input samples $x_1, \cdots, x_N$, an event A is defined by

$$\frac{1}{N}\sum_{i=1}^{N}\int_X |f_n(x) - f_n(x_i)| dP(x) - \frac{1}{N}\sum_{j=1}^{N}\frac{1}{N}\sum_{i=1}^{N}|f_n(x_j) - f_n(x_i)| \geqslant \varepsilon,$$

where the first and second terms represent the average over the expectation of $|f_n(x) - f_n(x_i)|$ and the unbiased estimator of the first term respectively.

Then, from the Hoeffding inequality, the probability of an event A is bounded by

$$\Pr\{A\} \leqslant \exp\left(\frac{-2\varepsilon^2 N}{\left(\max_{x\in X}\frac{1}{N}\sum_{i=1}^{N}|f_n(x) - f_n(x_i)|\right)^2}\right).$$

For the denominator in the argument of the exponent, we can consider the following inequality:

$$\begin{aligned}\max_{x\in X}\frac{1}{N}\sum_{i=1}^{N}|f_n(x) - f_n(x_i)| &\leqslant \frac{1}{N}\sum_{i=1}^{N}\max_{x\in X}|f_n(x) - f_n(x_i)| \\ &\leqslant \max_i \max_{x\in X}|f_n(x) - f_n(x_i)|.\end{aligned}$$

Let $x_i' = \arg\max_{x\in X}|f_n(x) - f_n(x_i)|$, $x_{i'} = \arg\min_j d(x_j, x_i')$, and $h_i = d(x_i', x_{i'})$ where $d(x,y)$ represents the distance measure defined by $d(x,y) = |x-y|$. Then,

$$\begin{aligned}\max_{x\in X}\frac{1}{N}\sum_{i=1}^{N}|f_n(x) - f_n(x_i)| &\leqslant \max_i\left(|f_n(x_i') - f_n(x_{i'})| + |f_n(x_{i'}) - f_n(x_i)|\right) \\ &\leqslant \max_i\left(\omega(f_n, h_i) + |f_n(x_{i'}) - f_n(x_i)|\right) \\ &\leqslant \omega(f_n, h_0) + |f_n(x_0') - f_n(x_0)|,\end{aligned}$$

where $h_0 \in \{h_1, \ldots, h_N\}$ and $x_0, x_0' \in \{x_1, \ldots, x_N\}$ satisfy

$$\omega(f_n, h_i) + |f_n(x_i) - f_n(x_j)| \leqslant \omega(f_n, h_0) + |f_n(x_0) - f_n(x_0')| \quad \text{for } i, j = 1, \cdots, N.$$

For the illustration of this concept, refer to Figure 2.

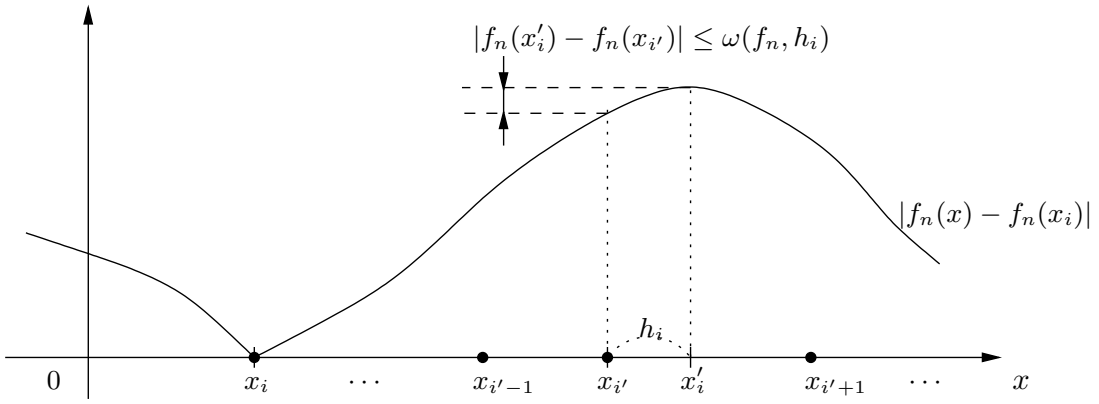


Figure 2: The plot of $|f_n(x) - f_n(x_i)|$ versus x: the value of $|f_n(x) - f_n(x_i)|$ is maximum at x_i' and this maximum value is decomposed by two factors: one is the value of $|f_n(x) - f_n(x_i)|$ at a sample point $x_{i'}$ and another is the modulus of continuity $\omega(f_n, h_i)$ with respect to h_i. The value h_i is chosen by the distance $d(x_i', x_{i'})$.

Thus, the probability of an event A is bounded by

$$\Pr\{A\} \leqslant \exp\left(\frac{-2\varepsilon^2 N}{(\omega(f_n, h_0) + |f_n(x_0) - f_n(x_0')|)^2}\right).$$

Here, let us set

$$\frac{\delta_1}{2} = \exp\left(\frac{-2\varepsilon^2 N}{(\omega(f_n, h_0) + |f_n(x_0) - f_n(x_0')|)^2}\right).$$

Then, with a probability of at least $1 - \delta_1/2$, we have

$$\frac{1}{N}\sum_{i=1}^{N}\int_X |f_n(x) - f_n(x_i)| dP(x) \leqslant \frac{1}{N^2}\sum_{i,j=1}^{N} |f_n(x_i) - f_n(x_j)| + \sqrt{\frac{1}{2N}\ln\frac{2}{\delta_1}}\left(\omega(f_n, h_0) + |f_n(x_0) - f_n(x_0')|\right). \quad (22)$$

On the other hand, for the target function f, we can apply a similar method. As a result, with a probability of at least $1 - \delta_1/2$, the following inequality holds:

$$\frac{1}{N}\sum_{i=1}^{N}\int_X |f(x) - f(x_i)| dP(x) \leqslant \frac{1}{N^2}\sum_{i,j=1}^{N} |f(x_i) - f(x_j)| + 2\|f\|_\infty \sqrt{\frac{1}{2N}\ln\frac{2}{\delta_1}}. \quad (23)$$

Let us consider the difference between the expected and empirical errors of $|f(x)-f_n(x)|$:

$$\begin{aligned}\int_X |f(x)-f_n(x)|dP(x) &- \frac{1}{N}\sum_{i=1}^{N}|f(x_i)-f_n(x_i)| \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_X (|f(x)-f_n(x)-f(x_i)+f_n(x_i)+f(x_i)-f_n(x_i)| \\ &\quad -|f(x_i)-f_n(x_i)|)dP(x) \\ &\leqslant \frac{1}{N}\sum_{i=1}^{N}\int_X |f(x)-f_n(x)-f(x_i)-f_n(x_i)|dP(x) \\ &\leqslant \frac{1}{N}\sum_{i=1}^{N}\int_X (|f(x)-f(x_i)|+|f_n(x)-f_n(x_i)|)dP(x).\end{aligned}$$

Then, from (22) and (23), the difference between the true and empirical risks is bounded by the following inequality with a probability of at least $1-\delta_1$:

$$\begin{aligned}\int_X |f(x)-f_n(x)|dP(x) - \frac{1}{N}\sum_{i=1}^{N}|f(x_i)-f_n(x_i)| & \\ \leqslant \frac{1}{N^2}\sum_{i,j=1}^{N}(|f(x_i)-f(x_j)|+|f_n(x_i)-f_n(x_j)|) & \\ + \sqrt{\frac{1}{2N}\ln\frac{2}{\delta_1}}\,(\omega(f_n,h_0)+|f_n(x_0)-f_n(y_0)|+2\|f\|_\infty). & \quad (24)\end{aligned}$$

Second, let us consider the noisy condition; that is, $y=f(x)+\varepsilon$. Here, we assume that for the output samples $y_1,\cdots,y_N$, the noise terms $\varepsilon_1,\ldots,\varepsilon_N$ are *i.i.d.* random variables with a mean of 0 and a variance of σ_ε^2. We will define the event B as

$$|\varepsilon| \geqslant a, \quad (25)$$

where a is a positive constant. Then, from the Chebyshev inequality,

$$\Pr\{B\} \leqslant \frac{\sigma_\varepsilon^2}{a^2}.$$

Let us set

$$\delta_2 = \frac{\sigma_\varepsilon^2}{a^2}.$$

Then, with a probability of at least $1-\delta_2$,

$$|\varepsilon| \leqslant \sigma_\varepsilon\sqrt{\frac{1}{\delta_2}}.$$

This implies that with a probability of at least $1-\delta_2$,

$$|y| \leqslant |f(x)|+|\varepsilon| \leqslant \|f\|_\infty + \sigma_\varepsilon\sqrt{\frac{1}{\delta_2}}.$$

Here, let us define the event E as

$$\frac{1}{N}\sum_{i=1}^{N}\int_{\mathbb{R}}|y-y_i|dP(y)-\frac{1}{N}\sum_{i=1}^{N}\frac{1}{N}\sum_{j=1}^{N}|y_j-y_i|>\varepsilon.$$

Then, from the Hoeffding inequality, we obtain

$$\begin{aligned}\Pr\{E|B^c\} &\leqslant \exp\left\{\frac{-2\varepsilon^2 N}{\left(\max_{y\in\mathbb{R}}\frac{1}{N}\sum_{i=1}^{N}|y-y_i|\right)^2}\right\}\\ &\leqslant \exp\left\{\frac{-\varepsilon^2 N}{2(\|f\|_\infty+\sigma_\varepsilon\sqrt{1/\delta_2})^2}\right\}.\end{aligned}$$

Let us set

$$\frac{\delta_1}{2}=\exp\left\{\frac{-\varepsilon^2 N}{2(\|f\|_\infty+\sigma_\varepsilon\sqrt{1/\delta_2})^2}\right\}.$$

Then, with a probability of at least $1-\delta_1/2-\delta_2$,

$$\frac{1}{N}\sum_{i=1}^{N}\int|y-y_i|dP(y)-\frac{1}{N^2}\sum_{i,j=1}^{N}|y_i-y_j|\leqslant\sqrt{\frac{2}{N}\ln\frac{2}{\delta_1}}\left(\|f\|_\infty+\sigma_\varepsilon\sqrt{\frac{1}{\delta_2}}\right)\tag{26}$$

since

$$\begin{aligned}\Pr\{E^c\} &\geqslant Pr\{E^c,B^c\}\\ &\geqslant Pr\{E^c|B^c\}Pr\{B^c\}\\ &\geqslant\left(1-\frac{\delta_1}{2}\right)(1-\delta_2)\\ &>1-\frac{\delta_1}{2}-\delta_2.\end{aligned}$$

Similar to (24), the difference between the expected and empirical risks of $|y-f_n(x)|$ is bounded by

$$\begin{aligned}\int_{X\times\mathbb{R}}|y-f_n(x)|dP(x,y) &- \frac{1}{N}\sum_{i=1}^{N}|y_i-f_n(x_i)|\\ &\leqslant \frac{1}{N}\sum_{i=1}^{N}\int_{X\times\mathbb{R}}|y-y_i|+|f_n(x)-f_n(x_i)|dP(x,y).\end{aligned}$$

Here, let us set $\delta_1=\delta_2=\delta$. This is possible by controlling the value of a in (25). Then, finally, from (22) and (26), with a probability of at least $1-2\delta$

$$\begin{aligned}\int_{X\times\mathbb{R}}|y-f_n(x)|dP(x,y) &- \frac{1}{N}\sum_{i=1}^{N}|y_i-f_n(x_i)|\\ &\leqslant \frac{1}{N^2}\sum_{i,j=1}^{N}(|y_j-y_i|+|f_n(x_j)-f_n(x_i)|)\\ &\quad+(\omega(f_n,h_0)+C)\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}},\end{aligned}\tag{27}$$

where $C=|f_n(x_0)-f_n(y_0)|+2\|f\|_\infty+2\sigma_\varepsilon\sqrt{1/\delta}$. □

A.4 Proof of Corollary 1

Let H_y be a matrix in which the ijth element is given by $|y_i - y_j|$ and an N dimensional vector $\mathbf{a}$ be given by

$$\mathbf{a} = \frac{1}{\sqrt{N}}(1, \cdots, 1)^T.$$

Then,

$$\frac{1}{N}\sum_{i,j=1}^{N} |y_i - y_j| = \mathbf{a}^T H_y \mathbf{a}. \tag{28}$$

Here, the matrix H_y can be decomposed by

$$H_y = E\Lambda E^T = \sum_{i=1}^{N} \lambda_i \mathbf{e}_i \mathbf{e}_i^T, \tag{29}$$

where E represents a matrix in which the ith column vector is the ith eigenvector $\mathbf{e}_i$ and Λ represents the diagonal matrix in which the ith diagonal element is the ith eigenvalue λ_i. Then, from (28) and (29),

$$\frac{1}{N}\sum_{i,j=1}^{N} |y_i - y_j| = \sum_{i=1}^{N} \lambda_i (\mathbf{a}^T \mathbf{e}_i)^2 \leqslant \max_i \{\lambda_i\}.$$

Now, let us consider the following inequality:

$$\begin{aligned}
\frac{1}{N^2}\sum_{i,j=1}^{N} |f_n(x_i) - f_n(x_j)| &\leqslant \frac{1}{N^2}\sum_{i,j=1}^{N} |f_n(x_i) - y_i| \\
&\quad + \frac{1}{N^2}\sum_{i,j=1}^{N} |y_i - y_j| + \frac{1}{N^2}\sum_{i,j=1}^{N} |y_j - f_n(x_j)| \\
&= 2R_{emp}(f_n)_{L_1} + \frac{1}{N^2}\sum_{i,j=1}^{N} |y_i - y_j| \\
&\leqslant 2R_{emp}(f_n)_{L_1} + \frac{1}{N}\max_i \{\lambda_i\}.
\end{aligned}$$

Therefore, from the above inequality and (27), the following inequality holds with a probability of at least $1 - 2\delta$:

$$R(f_n) \leqslant 3R_{emp}(f_n) + \frac{2}{N}\max_i \{\lambda_i\} + (\omega(f_n, h_0) + C)\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}.$$

□

Appendix B.

In this appendix, we prove the theorem 2 in Section 3.2; that is, the case of multivariate target functions.

B.1 Proof of Theorem 2

First, let us consider noise free target function; that is,

$$y = f(\mathbf{x}).$$

The probability that the difference between the expected and empirical risks is larger than a positive constant ε can be described by

$$\Pr\{R(f_n)_{L_1} - R_{emp}(f_n)_{L_1} > \varepsilon\} \leqslant \exp\left\{\frac{-2\varepsilon^2 N}{(\max_{\mathbf{x}\in X}|y - f_n(\mathbf{x})|)^2}\right\} \tag{30}$$

from the Hoeffding inequality (Hoeffding, 1963). Here, there exist $\mathbf{x}_0 \in X$ and $\mathbf{x}_{i_0} \in \{\mathbf{x}_0, \cdots, \mathbf{x}_N\}$ such that

$$\mathbf{x}_0 = \arg\max_{\mathbf{x}\in X} |f(\mathbf{x}) - f_n(\mathbf{x})| \text{ and } d(\mathbf{x}_{i_0}, \mathbf{x}_0) \leqslant h_0$$

because $f - f_n \in C(X)$ and X is a compact subset of $\mathbb{R}^m$. Thus, from the dominator term of the righthand side of (30), we have

$$\begin{aligned}\max_{\mathbf{x}\in X} |f(\mathbf{x}) - f_n(\mathbf{x})| &\leqslant |f(\mathbf{x}_0) - f_n(\mathbf{x}_0) - f(\mathbf{x}_{i_0}) + f_n(\mathbf{x}_{i_0})| + |f(\mathbf{x}_{i_0}) - f_n(\mathbf{x}_{i_0})| \\ &\leqslant \omega(f - f_n, h_0) + |f(\mathbf{x}_{i_0}) - f_n(\mathbf{x}_{i_0})|. \end{aligned} \tag{31}$$

Here, we set the bound on the probability of (30) as

$$\begin{aligned}\exp\left\{\frac{-2\varepsilon^2 N}{(\max_{\mathbf{x}\in X}|f(\mathbf{x}) - f_n(\mathbf{x})|)^2}\right\} &\leqslant \exp\left\{\frac{-2\varepsilon^2 N}{(\omega(f - f_n, h_0) + |f(\mathbf{x}_{i_0}) - f_n(\mathbf{x}_{i_0})|)^2}\right\} \\ &\leqslant \frac{\delta}{2}. \end{aligned} \tag{32}$$

Therefore, from (30), (31), and (32), the following inequality holds with a probability of at least $1 - \delta/2$:

$$R(f_n)_{L_1} \leqslant R_{emp}(f_n)_{L_1} + \{\omega(f - f_n, h_0) + |f(\mathbf{x}_{i_0}) - f_n(\mathbf{x}_{i_0})|\}\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}. \tag{33}$$

Second, let us consider the noisy target function; that is,

$$y = f(\mathbf{x}) + \varepsilon.$$

From Chebyshev inequality, the following inequality always holds:

$$\Pr\{|\varepsilon| > a\} \leqslant \frac{\sigma_\varepsilon^2}{a^2}, \tag{34}$$

where a represents a positive constant. In this case, from the triangular inequality, $|y - f_n(\mathbf{x})|$ has the following upper bound:

$$\max_{\mathbf{x}\in X} |y - f_n(\mathbf{x})| \leqslant \max_{\mathbf{x}\in X} |f(\mathbf{x}) - f_n(\mathbf{x})| + |\varepsilon|. \tag{35}$$

Let us set the bound on the probability of (34) as

$$\frac{\sigma_\varepsilon^2}{a^2} = \frac{\delta}{2}. \tag{36}$$

Then, from (31), (35), and (36), the following inequality holds with a probability of at least $1-\delta/2$:

$$\begin{aligned}\max_{\mathbf{x}\in X}|y-f_n(\mathbf{x})| &\leqslant \max_{\mathbf{x}\in X}|f(\mathbf{x})-f_n(\mathbf{x})|+\sigma_\varepsilon\sqrt{\frac{2}{\delta}} \\ &\leqslant \omega(f-f_n,h_0)+|f(\mathbf{x}_{i_0})-f_n(\mathbf{x}_{i_0})|+\sigma_\varepsilon\sqrt{\frac{2}{\delta}}. \end{aligned} \tag{37}$$

Therefore, from (33) and (37), the following inequality holds with a probability of at least $1-\delta$:

$$R(f_n)_{L_1} \leqslant R_{emp}(f_n)_{L_1}+\left\{\omega(f-f_n,h_0)+|f(\mathbf{x}_{i_0})-f_n(\mathbf{x}_{i_0})|+\sigma_\varepsilon\sqrt{\frac{2}{\delta}}\right\}\sqrt{\frac{1}{2N}\ln\frac{2}{\delta}}.$$

□

References

H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proceedings of the Second International Symposium on Information Theory*, pages 267–281, 1973.

G. Anastassiou and S. Gal. *Approximation Theory: Moduli of Continuity and Global Smoothness Preservation*. Birkhäuser, Boston, 2000.

A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44:2743–2760, 1998.

C. Chang and C. Lin. LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

O. Chapelle, V. Vapnik, and Y. Bengio. Model selection for small sample regression. *Machine Learning*, 48:315-333, 2002.

V. Cherkassky and Y. Ma. Comparison of model selection for regression. *Neural Computation*, 15:1691–1714, 2003.

V. Cherkassky, X. Shao, F. Mulier, and V. Vapnik. Model complexity control for regression using VC generalization bounds. *IEEE Transactions on Neural Networks*, 10:1075–1089, 1999.

S. Cohen and N. Intrator. On different model selection criteria in a forward and backward regression hybrid network. *International Journal of Pattern Recognition and Artificial Intelligence*, 18:847–865, 2004.

D. Donoho and I. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:1200–1224, 1995.

D. Foster and E. George. The risk inflation criterion for multiple regression. *Annals of Statistics*, 22:1947–1975, 1994.

T. Hastie, R. Tibshirani, and J. Friedman. Note on "comparison of model selection for regression" by V. Cherkassky and Y. Ma. *Neural Computation*, 15:1477–1480, 2003.

G. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory*, pages 5–13, 1993.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13-30, 1963.

M. Karpinski and A. Macintyre. Polynomial bounds for VC dimension of sigmoidal neural networks. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 200–208, 1995.

I. Koo and R. Kil. Nonlinear model selection based on the modulus of continuity. In *Proceedings of World Congress on Computational Intelligence*, pages 3552–3559, 2006.

G. Lorentz. *Approximation of Functions*. Chelsea Publishing Company, New York, 1986.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

A. Sakurai. Polynomial bounds for the VC dimension of sigmoidal, radial basis function, and sigma-pi networks. In *Proceedings of the World Congress on Neural Networks*, pages 58–63, 1995.

G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

A. Timan. *Theory of Approximation of Functions of a Real Variable*. English translation 1963, Pergaman Press, Russian original published in Moscow by Fizmatgiz in 1960.

V. Vapnik. *Statistical Learning Theory*. J. Wiley, 1998.

G. Wahba, G.Golub, and M. Heath. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

Journal of Machine Learning Research 9 (2008) 2635-2675 Submitted 7/08; Published 12/08

Multi-Agent Reinforcement Learning in Common Interest and Fixed Sum Stochastic Games: An Experimental Study*

Avraham Bab BAB@CS.BGU.AC.IL
Ronen I. Brafman BRAFMAN@CS.BGU.AC.IL
Department of Computer Science
Ben-Gurion University
Beer-Sheva 84105, Israel

Editor: Michael Littman

Abstract

Multi Agent Reinforcement Learning (MARL) has received continually growing attention in the past decade. Many algorithms that vary in their approaches to the different subtasks of MARL have been developed. However, the theoretical convergence results for these algorithms do not give a clue as to their practical performance nor supply insights to the dynamics of the learning process itself. This work is a comprehensive empirical study conducted on *MGS*, a simulation system developed for this purpose. It surveys the important algorithms in the field, demonstrates the strengths and weaknesses of the different approaches to MARL through application of FriendQ, OAL, WoLF, FoeQ, Rmax, and other algorithms to a variety of fully cooperative and fully competitive domains in self and heterogeneous play, and supplies an informal analysis of the resulting learning processes. The results can aid in the design of new learning algorithms, in matching existing algorithms to specific tasks, and may guide further research and formal analysis of the learning processes.

Keywords: reinforcement learning, multi-agent reinforcement learning, stochastic games

1. Introduction

Multi-Agent Reinforcement Learning (MARL) deals with the problem of learning to behave well through trial and error interaction within a multi-agent dynamics environment when the environmental dynamic and the algorithms employed by the other agents are initially unknown. Potential applications of MARL range from load balancing in networks (Schaerf et al., 1995) and e-commerce (Sridharan and Tesauro, 2000) to planetary exploration by mobile robot teams (Zheng et al., 2006).

MARL adopts the game theory model of a Stochastic (a.k.a. Markov) Game (SG) to model the multi-agent-environment interaction. The non-cooperative[1] game theoretic solution concept for SGs is the Nash Equilibrium (NE). A NE is a behavioral profile, namely a set of decision rules, or policies, for all agents, such that no agent can benefit from unilaterally changing its behavior. However, SGs may have multiple NEs with different values, none of which is necessarily strictly optimal (i.e., preferable by all agents to all other NEs). Thus, in the general case, it is not clear which behavior should be considered "optimal," even when the environmental dynamics and the other players' set of possible strategies are known. For this reason, development of MARL algorithms

*. A preliminary version of this paper that covered some of the results on common-interest games appeared in Bab and Brafman (2004).

1. In this context, the meaning of 'non-cooperative' is that agents are selfish and do not collaborate or communicate with other agents, except through the game.

has concentrated on algorithms for classes of SGs in which there is a unique NE, or in which all NEs have the same value. In such cases, it is possible to measure the performance of learning algorithms against a well defined target.[2] In particular, most MARL algorithms are shown to converge to such NEs in self play in either Common Interest SGs (CISGs) or Fixed Sum SGs (FSSGs), which we describe next.

CISGs model environments in which the agents share common interests and have no conflicting interests. In such environments, defining an optimal joint behavior for all agents is straightforward—it is the joint behavior that maximizes the common interests. However, since the agents are independent, they face the task of coordinating such joint behavior in the case in which there are several optimal options. FSSGs, on the other hand, model environments in which two agents have fully conflicting interests. In FSSGs, there is a well defined *minimax* solution (Filar and Vrieze, 1997).

Several different MARL algorithms have been proved to converge in the limit to optimal behavior in CISGs (Littman, 2001; Wang and Sandholm, 2002) and in FSSGs (Littman, 1994). One has been shown to converge to ε-optimal behavior in polynomial time in both CISGs and FSSGs (Brafman and Tennenholtz, 2002, 2003). Since MARL is, by its nature, an online task, determining the abilities of the algorithms in practical domains is important. However, existing theoretical results tell us very little about the practical efficacy of the algorithms;[3] to this end a comprehensive empirical comparison is necessary. Experimental results that have been published in the literature on CISGs (Claus and Boutilier, 1997; Wang and Sandholm, 2002; Chalkiadakis and Boutilier, 2003) and on FSSGs (Littman, 1994; Uther and Veloso, 2003; Bowling and Veloso, 2002), do not meet this demand. They do not examine representative samples of algorithms and/or use small and simple test models and/or do not examine online learning. Furthermore, the different experimental setups used in different publications do not enable cross comparisons of the algorithms they examine.

This work provides a comprehensive empirical study of MARL algorithms in CISGs and FSSGs. It offers a decomposition of the MARL task into subtasks. It then compares three algorithms for learning in CISGs: FriendQ (Littman, 2001), OAL (Wang and Sandholm, 2002), and Rmax (Brafman and Tennenholtz, 2002); and three algorithms for learning in FSSGs: FoeQ (Littman, 1994, 2001), WoLF (Bowling and Veloso, 2002) and Rmax (Brafman and Tennenholtz, 2002). These algorithms were selected because they represent a variety of approaches to the offered subtasks, while providing certain convergence guarantees. We experimented with diverse variants of these algorithms on several non-trivial test environments which we designed to demonstrate the efficacy of the different approaches in each of the subtasks. To concentrate attention on the basic learning task, full state observability and perfect monitoring (that is, the ability to observe the actions of other agents) are assumed. The results allow us to rank the performance of the algorithms according to properties of the environment and possible performance measures.

The experiments for this work have been conducted using MGS, a Markov Game Simulation system developed for this purpose. MGS is implemented in the Java programming language and supplies interfaces and abstract classes for the simple creation of players and grid worlds and convenient logging. We believe that MGS can be of good service to both MARL algorithm designers and users. MGS is free, open source software available at `http://www.cs.bgu.ac.il/~mal`.

2. Much recent work is concerned with the question of how to define and evaluate the performance of learning algorithms in more general games. See, for example, Vohraa and Wellman (2007) which is devoted to this issue.
3. Vidal and Durfee (2003) take a step towards theoretical analysis of the learning dynamics. They offer theoretical tools to analyzing and predicting behavior of multi-agent systems that are represented by simpler models than SGs. Powers and Shoham (2005) offer experimental results on iterative games, which are a much simpler model than SGs.

The paper is organized as follows. Necessary background is given in Section 2. Sections 3 and 4 describe the particular problems and algorithms for CISGs and FSSGs, respectively, and present experimental results and analysis. Section 5 describes MGS and Section 6 concludes the paper.

2. Multi-Agent Reinforcement Learning and Stochastic Games

Multi-Agent Reinforcement Learning (MARL) is an extension of RL (Sutton and Barto, 1998; Kaelbling et al., 1996) to multi-agent environments. It deals with the problems associated with the learning of optimal behavior from the point of view of an agent acting in a multi-agent environment. At the outset, the environmental dynamics and the algorithms employed by the other players are unknown to the given agent. The environment is modeled by a finite set of states and the agents-environment interaction is discretized into time steps. At each time step, the players simultaneously choose actions, available from individual sets of actions. Depending stochastically on the joint action, the environment transitions into its next state and each player is rewarded. The present work assumes full state observability and perfect monitoring, namely, the agent observes the actions taken and rewards received by the other players. It also assumes that the agents have no additional means of communication. The multi-agent-environment interaction is modeled by a Stochastic (a.k.a Markov) Game (SG).

Definition 2.1 (Stochastic Game) *An SG* $G := \{\alpha, A, S, T, R\}$ *consists of:*

- $\alpha = \{1, ..., n\}$ *- a set of players. We will typically use n to denote the number of players.*
- $A = A_1 \times A_2 \times ... \times A_n$ *– a set of joint actions.* A_i *is a set of private actions available to player i.*
- S *- a set of states.*
- $T : S \times A \times S \to [0,1]$ *- a transition function.* $T(s,a,s') = Pr(s' \mid s,a)$ *is the probability that the system transitions to state* s' *when joint action a is taken at state s (*$\sum_{s'} T(s,a,s') = 1$*).*
- $R : S \times A \times S \to \mathbb{R}^n$ *- a payoff function.* $[R(s,a,s')]_i$ *is i's reward upon transition from state s to state* s' *under joint action a.*

The behavior of player i in an SG is described by a *policy*. A policy is a mapping $\pi_i : \mathscr{H} \to \mathscr{PD}(A_i)$ where $\mathscr{H} := \{(s_0, a_1, s_1, a_2, ..., s_j) \mid j \geq 0\}$ is the set of possible histories of the process and $\mathscr{PD}(A_i)$ is a probability distribution over A_i. A policy that depends only on the current state of the process, that is, $\pi_i : S \to \mathscr{PD}(A_i)$ is called *stationary*. A deterministic policy, that is a mapping, $\pi_i : \mathscr{H} \to A_i$ is called *pure*, whereas a stochastic policy is called *mixed*. A tuple of policies $\pi = (\pi_1, \ldots, \pi_n)$ for n players of a SG is called a *policy profile*. The objective of an agent in a SG is to maximize some function of its accumulated payoffs, referred to as the agent's return. In this study, the infinite horizon discounted return (IHDR) is considered. The expected IHDR for player i, resulting from policy profile π, is defined by the sum $\sum_{t=0}^{\infty} \gamma^t E^{\pi}(r_t^i)$ where r_t^i is player i's payoff at time t and $\gamma \in [0,1)$ is a discount factor. Consequently, a state-policy value function, V is defined by $V_i(s,\pi) = \sum_{t=0}^{\infty} \gamma^t E^{\pi}(r_t^i \mid s_0 = s)$.

We note that different algorithms optimize different objectives. Yet, typically, the same underlying ideas can be used to formulate different variants of the same basic algorithm that aim to

maximize different natural objectives. While we use formulations that aim to maximize IHDR, the games we experiment on are such that any good policy will reach an absorbing state (following which the agents are placed in their initial states) quickly. In this setting, given a reasonably high discount factor, γ, IHDR maximizing behavior will be identical to behavior maximizing average reward. Consequently, we will sometimes find it more natural to report performance measures such as average reward per step.

For single agent domains, where $n = 1$, there is always an optimal pure stationary policy that maximizes $V(s,\pi)$ for all $s \in S$ (Filar and Vrieze, 1997). The single-agent state-policy value function for the optimal policy, referred to as the state-value function, is the unique fixed point of the Bellman optimality equations

$$V^*(s) = \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right), \forall s \in S \, .$$

An optimal policy may be specified by $\pi^*(s) = arg\max_{a \in A}(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s'))$ (Puterman, 1994). Many single agent Reinforcement Learning (RL) methods interleave approximation of the value function with derivation of a learning policy from the current approximation.

In MARL, maximizing the IHDR cannot be done by simply maximizing over (private) policies since the return depends also on the other players' policies which, in turn, may depend on the agent's actions. Hence, to maximize the IHDR, the agent must adopt a policy that is a *best response* to the other players' policies. Formally, π_i is a best response to $\pi^{-i} = (\pi_1, ..., \pi_{i-1}, \pi_{i+1}, ..., \pi_n)$ if $V_i(s, \pi_1, ..., \pi_i, ..., \pi_n) \geq V_i(s, \pi_1, ..., \pi_i', ..., \pi_n)$ for all π_i' and $s \in S$. A best response function is defined by $BR(\pi^{-i}) = \left\{ \pi_i \mid \pi_i \textit{ is a best response to } \pi^{-i} \right\}$. In general, $\bigcap_{\pi^{-i}} BR(\pi^{-i}) = \emptyset$, namely, there is no policy that is a best response to all of the possible behaviors of the other players.

Whereas the goal of single-agent reinforcement learning is clear—maximizes some aggregate of your reward stream, the picture is more complex in multi-agent settings. Here, one's performance depends on what the other agents do, and strategic considerations come to the fore. For instance, the well-known notion of Nash Equilibria does not, in general, provide a clear target for learning algorithms, as many such equilibria may exist in a game, none of which dominates the others. Although some recent work has attempted to clarify this issue (Brafman and Tennenholtz, 2004; Shoham et al., 2007), there is still no clear agreement on the goal of MARL. However, there are two special classes of SGs in which there is a clear target for learning: Common-interest SGs, and Fixed-sum SGs. These are two extreme cases of SGs where players are either fully cooperative or fully opposed. Much work in the area of MARL has concentrated on these classes of SGs, and algorithms with good theoretical guarantees exist for each of them. In this paper, we analyze a number of algorithms for such games.

3. Learning in CISGs

In CISGs, the payoffs are identical for all agents. That is, for any given choice of s, a and s' and any pair i, j of agents, we have that $[R(s,a,s')]_i = [R(s,a,s')]_j$. Therefore, all agents have identical interests and we may speak of optimal joint policies, namely, policy profiles that maximize the common IHDR for the team of agents. Such profiles are also NEs because no agent can gain by deviating from them. CISGs pose all the standard challenges of single-agent RL, in particular the need to balance exploration and exploitation and to propagate new experience. In addition, they challenge the agents to coordinate behavior since to obtain maximum value may require that agents

select a particular joint action. On the other hand, CISGs do not require that agents confront the more difficult task of optimizing behavior against an adversary.

For efficient learning in CISGs, agents are required to coordinate on two levels: (i) select whether to explore or exploit in unison; and (ii) coordinate the exploration and exploitation moves. This requirement stems from the dependence of the team's next state on the actions of *all* its members. Hence, it is impossible for the team to exploit unless all agents exploit together, and using the same choice of exploitation strategy. Exploration, too, can be less effective when only some agents explore.

Furthermore, even when the model is known, multiple NEs yielding maximal payoffs to the agents are likely to exist, and the agents still face the task of reaching consensus on which specific NE to play.

This section describes and compares three algorithms for learning in CISGs: OAL (Wang and Sandholm, 2002), FriendQ (Littman, 2001), and Rmax (Brafman and Tennenholtz, 2002). They were selected because each embodies a different approach to learning, while guaranteeing convergence to optimal behavior in CISGs. Diverse variants of these algorithms are examined with the aim of gaining better understanding of their performance with respect to their approach to exploration-exploitation, information propagation, and coordination tasks.[4] These variants and the tasks on which they were tested are described in the following subsections.

3.1 FriendQ

FriendQ (Littman, 2001) extends single agent Q-learning into CISGs. After taking a joint action $a = (a_1, ..., a_n)$ in state s at time t and reaching state s' with reward r_{cur}, each agent updates its Q-value estimates for $\langle s, a \rangle$ as follows:

$$Q_t(s,a) \leftarrow (1-\alpha_t)Q_{t-1}(s,a) + \alpha_t \left(r_{cur} + \gamma \max_{a' \in A} Q(s', a') \right).$$

As in single agent Q-learning, given that $\sum_{t=0}^{\infty} \alpha_t = \infty$, $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$ and that every joint action is performed infinitely often in every state, the Q-values are guaranteed to converge asymptotically to Q^* (Littman, 2001). Convergence to optimal behavior is achieved using *Greedy in the Limit with Infinite Exploration Learning Policies* (GLIELP) (Sutton and Barto, 1998).

There are two types of GLIELPs, *directed* and *undirected*. Directed GLIELPs reason about the uncertainty of the current belief about action values (Kaelbling, 1993; Dearden et al., 1998, 1999; Chalkiadakis and Boutilier, 2003). However, the computational complexity of the underlying statistical methods makes directed exploration impractical for simulations of the size conducted in this study.[5] Two popular undirected exploration methods are ε-greedy action selection and Boltzman distributed action selection. There is no established technique for applying Boltzman exploration to FriendQ, so in our experiments it is executed with ε-greedy exploration only. ε-greedy exploration is applied to SGs in the following way: each agent randomly picks an exploratory private action with probability ε, and with probability $1-\varepsilon$ takes its part of an optimal (greedy) joint action with

4. By this we mean the ability of the algorithm to propagate information observed in one state to other states. For example, Q-learning does not propagate information beyond the current state, unless techniques such as eligibility traces are used.

5. It can be argued that many realistic applications impose severe constraints on the length of trajectories. In this case, directed exploration techniques and techniques such as transfer learning appear to be essential for success. Conducting a study of algorithms for such contexts would seem to be of great interest.

respect to the current Q-value (Claus and Boutilier, 1997). ε is asymptotically decreased to zero over time.

Since full state observability, perfect monitoring, and identical initial Q-values to all agents are assumed, all agents maintain identical Q-values throughout the process, and consequently the same classification of greedy actions. But, two problems arise: (i) Because randomization is used to select exploration *or* exploitation, the agents cannot coordinate their choice of when and what to explore. (ii) In the case of multiple optimal policies, that is, several joint actions with maximal Q-values in a certain state, the agents must agree on one such action. The original FriendQ algorithm has no explicit mechanism for handling these issues.

This work compares some enhanced versions of FriendQ: First, Uncoordinated FriendQ (UFQ), the simple version described above, is tested. Next, the effect of adding coordination of greedy joint actions by using techniques introduced by Brafman and Tennenholtz (2003) is examined. Basically, a shared order over joint actions is used for selecting among equivalent NEs. If such an order is not built into the agents, it is established during a preliminary phase using an existing technique (Brafman and Tennenholtz, 2003). This version is referred to as Coordinated FriendQ (CFQ). Then, coordination of exploration and exploratory actions is added in Deterministic FriendQ (DFQ). In DFQ, the agents explore and exploit in unison, always exploring the least tried joint action. An exploratory action is taken each $\lfloor 1/\varepsilon \rfloor' th$ move. Finally, we add Eligibility Traces (Sutton and Barto, 1998) to DFQ (ETDFQ).[6] Eligibility Traces propagate new experience to update Q-values of previously visited states and not only the most recently visited state.

3.2 OAL

OAL combines classic *model-based* reinforcement learning with a new *fictitious play* algorithm for action and equilibrium selection named BAP (Biased Adaptive Play) (Wang and Sandholm, 2002). BAP is an action-selection method for a class of repeated games that contains common interest games. Here, BAP is described in the context of common interest repeated games. Let m and k be integers such that $1 \leq k \leq m$. Each agent maintains a memory of the past m joint actions. At the first m steps of the repeated game, each player randomly chooses its actions. Starting from step $m+1$, each agent randomly samples k out of the m most recent joint actions. Let SP_i be the set of k joint actions drawn by agent i at some time step. If (i) there is a joint action a' that is estimated to be ε-optimal, such that for all $a \in SP_i$, $a^{-i} \subset a'$ (where $a^{-i} \subset a'$ denotes the fact that the individual actions of all agents other than agent i are identical in a and a'), and (ii) there is at least one optimal joint action $a \in SP_i$, then agent i chooses its part of the most recent optimal joint action in SP_i. If the above two conditions are not met, then agent i chooses an action a_i that maximizes its expected payoff under the assumption that the other players' sampled history reflects their future behavior. This type of action selection is known as *fictitious play* (Brown, 1951).

$$EP(a_i) = \sum_{a^{-i} \in SP_i} R(a_i \cup a^{-i}) \frac{N(a^{-i}, SP_i)}{k}$$

where $N(a^{-i}, SP_i)$ is the number of occurrences of a^{-i} in SP_i. Given that there is no sub-optimal NE and $m \geq k(n+2)$, BAP is guaranteed to converge to a NE. It was shown that, for every game that satisfies these conditions, there is some positive probability p and some positive integer T such that

6. A variant of Eligibility Traces called Replacing Traces was used (Singh and Sutton, 1996). In Replacing Traces, the eligibility traces are bounded by 1.

for any history of plays, with probability at least p, BAP converges to a consensus in T steps. That is, all players agree in the same joint-action which is a NE.

After observing a transition from state s on action a, OAL updates Q-values according to the learning rule

$$Q_{t+1}(s,a) = R_t(s,a) + \gamma \sum_{s'} T_t(s,a,s') \max_{a'} Q_t(s',a')$$

where R_t, the approximated mean reward, and T_t, the approximated transition probability are estimated using the statistics gathered up to time t. At each step, OAL constructs a Virtual Game (VG) for the current state-game (the matrix game defined by the current state's Q-values) and plays according to it. The VG has common payoff 1 for any optimal joint action and payoff 0 for any other action. In our implementation we use the VG in conjunction with ε-greedy as well as Boltzman action selection. Boltzman action selection is implemented as follows: At each step, an action is sampled according to the Boltzman distribution induced by the Expected Payoffs in the current VG

$$\frac{e^{EP(s,a)/\tau}}{\sum_b e^{EP(s,b)/\tau}} .$$

If a sub-optimal action is sampled, it is explored by the agent, otherwise BAP is executed on the VG to select an exploitation action.

We examine OAL also with an addition of Prioritized Sweeping (PS) (Moore and Atkeson, 1993) to the underlying Q-learning algorithm (PSOAL). PS is a heuristic method for optimizing finite propagation of TD-errors in the model. PS attempts to order propagation according to the size of the change to the Q-values, for example, states that are liable to have a greater update should be updated first. For comparison, a combination of the model-based Q-learning algorithm used by OAL with the action and equilibrium-selection technique used by CFQ is also examined. This combination is referred to as ModelQ (MQ).

3.3 Rmax

Rmax (Brafman and Tennenholtz, 2002) is a model-based algorithm designed to handle learning in MDPs and in fixed-sum stochastic games. However, because Rmax does not make random decisions (e.g., random exploration), its MDP version can also be used to tackle MARL in CISGs. Brafman and Tennenholtz (2003) view a CISG as an MDP controlled by a distributed team of agents and show how such a team can coordinate its behavior given a deterministic algorithm such as Rmax. In a preliminary phase of the game, a protocol is used to establish common knowledge of the individual action sets, of orders over these sets, and of an order over the agents. At each point in time, all agents have an identical model of the environment and know what joint action needs to be executed next (when a number of actions are optimal with respect to the current state, the agents use the shared order over joint actions to select among these actions). Thus, each agent plays its part of this action. It is shown that even weaker coordination devices can be used, and that these ideas can be employed even under imperfect monitoring.

Rmax maintains a model of the environment, initialized in a particular optimistic manner. It always behaves optimally with respect to its current model, while updating this model (and hence its behavior) when new observations are made. The model M' used by Rmax consists of $n+1$ states $S' = \{s_0, ..., s_n\}$ where $s_1, ..., s_n$ correspond to the states of the real model M, and s_0 is a fictitious state.[7] The transition probabilities in M' are initialized to $T_{M'}(s,a,s_0) = 1 \ \forall \langle s,a \rangle \in S' \times A$. The

7. The model may be constructed online as states are discovered.

reward function is initialized to $R_{M'}(s,a) = R_{max}\ \forall \langle s,a\rangle \in S' \times A$, where R_{max} is an upper bound on $\max_{s\in S, a\in A} R(s,a)$. Each state/joint-action pair in M' is classified either as *known* or as *unknown*. Initially, all entries are unknown.

Rmax computes an optimal policy with respect to M' and follows this policy until some entry becomes known. It keeps the following records: (i) number of times each action was taken at each state and the resulting state; (ii) the actual rewards, $r_{ac}(s,a)$, received at each entry. An entry (s,a) becomes known after it has been sampled $K1$ times, such that with high probability $T_M(s,a,s') - \rho \leq PE(s,a,s'|K1) \leq T_M(s,a,s') + \rho$ where T_M is the transition function in M, $PE(s,a,\cdot\ |K1)$ is the empirical transition probability according to the $K1$ samples, and ρ is the accuracy required from M'. When an entry (s,a) becomes known, the following updates are made: $T_{M'}(s,a,\cdot) \leftarrow PE(s,a,\cdot\ |K1)$ and $R_{M'}(s,a) \leftarrow r_{ac}(s,a)$. Then, a new deterministic optimal policy with respect to the updated model is computed and followed. Rmax converges to an ε-optimal policy in polynomial number of steps.

The worst-case bounds on $K1$ (Brafman and Tennenholtz, 2002) assume maximal entropy on the transition probabilities, that is, $T_M(s,a,s') = 1/|S|$ for all s,a,s'. These bounds, although polynomial, are impractical. In the experiments, these bounds are violated, which enables us to eliminate knowledge of the state space size. Furthermore, R_{max} is not assumed to be known. Instead, it is initialized to some positive value and updated online to be twice the highest reward encountered so far.

3.4 Discussion of Algorithms

Returning to the FriendQ algorithm, the efficiency of GLIELPs depends on the topology and dynamic of the environment. If the probability to explore falls low before "profitable" parts of the environment are sufficiently sampled, the increasing bias to exploit may keep the agents in suboptimal states. As a result, GLIELPs can exhibit significant differences depending on the particular schedule of exploration. In model free algorithms, and FriendQ, in particular, this phenomenon is intensified by the decreasing learning rate that makes learning from the same experience slower over time. GLIELPs also suffer from their inability to completely stop exploration at some point. Thus, even when greedy behavior is optimal, the agent is unable to attain optimal return.

The exploration method of Rmax is less susceptible to the structure of the environment. As long as Rmax cannot achieve actual return ε-close to optimal, it will have a strong bias for exploration since unknown entries seem very attractive. This strategy is profitable when the model can be learned in a short time. However, the theoretical worst-case bounds for convergence in Rmax are impractical. In practice, much lower values of $K1$ suffice. Bayesian exploration (Dearden et al., 1999; Chalkiadakis and Boutilier, 2003) and locality considerations might help to obtain better adaptive bounds, but these approaches are not pursued here.

GLIELPs make learning "slower" as the agents get "older". To accelerate learning, an algorithm can try to use new experience in a more exhaustive manner, using it to improve behavior in previously visited states. Eligibility traces are used to propagate information in FriendQ. In model-based algorithms, an exhaustive computation per new experience is too expensive (in CPU time). Thus, OAL is tested with Prioritized Sweeping and Rmax makes one exhaustive computation each time a new entry becomes known (and does no further computation).

Exploration in FriendQ and OAL algorithms is not coordinated. Each of the agents independently chooses an exploratory action with some diminishing probability. Thus, joint actions that

have no element (private action) of some optimal joint action have a lower chance of being explored. Hence, some popular techniques for decreasing exploration in the single agent case lead to finite exploration in the multi agent case. For example, taking $\varepsilon = 1/time$ for ε-greedy policies will make the chance of exploring such joint actions $1/time^n$, where *n* is the number of agents.

Equilibrium selection in Rmax and CFQ comes with no cost. In OAL, it is essentially a random protocol for achieving consensus. This protocol may take long to reach consensus with respect to the current Q-values, but provides for another exploration mechanism at early stages, when Q-values are frequently updated.

All three algorithms have parameters that need to be preset. Parameter tuning is task specific and based more on intuition and trial and error than on theoretical results. FriendQ has a range of parameters for decaying the learning rate, the exploration probability and the eligibility traces, which also pose inter-parameter dependencies. For decreasing the learning rate parameter, we used the results presented in Even-Dar and Mansour (2003). OAL takes parameters for history sample size and for exploration. In this respect, Rmax is friendlier. It has a single and very intuitive parameter—number of visits to declare an entry *known*. When the value of this parameter is high, a very accurate model is learned and behavior will be, eventually, very close to optimal. But this comes at the cost of possibly unnecessary exploration and delayed exploitation.

Table 1 summarizes the differences between the three algorithms according to the features mentioned above.

property	**UFQ**	**CFQ**	**DFQ**	**OAL**	**Rmax**
Exploration	ε-greedy with exponential and polynomial decay of ε			Boltzman & ε-greedy (polynomial decay)	Greedy w.r.t optimistic model
Coordination	None	Common order; non-deterministic	Common order; deterministic	Random protocol; non-deterministic	Common order; deterministic
Greedy Action Selection	Maximize common return			Fictitious play/consensus	Maximize common return
Information Propagation	Single sweep per step & ET			Single sweep per step & PS	Limited exhaustive computations
Parameter Tuning	Many parameters, task sensitive, not intuitive				One parameter, not sensitive, intuitive

Table 1: Major differences between the experimented algorithms.

3.5 Experimental Results & Analysis

This section describes experiments with the FriendQ, OAL, and Rmax algorithms on three CISGs. The games were designed to evaluate the effects of exploration, coordination, and information-propagation methods on performance in different environments. All games are grid-based. The grid cells are referred to by *(row, column)* coordinates indexed from $(0,0)$ at the top left corner of the grid. In all games, the available actions for each agent are *up*, *down*, *left*, *right*, and *stand*. The games were played in both deterministic and stochastic modes. In deterministic mode, the action always succeeds. In stochastic mode, each action, excluding *stand*, succeeds with probability 0.6. With probability 0.4, uniformly at random, the agent is moved to the each of the other adjacent cells or left in place. Action *stand* succeeds with probability 1. If the direction of motion is towards a wall, the player remains in place. Similarly, two players cannot occupy the same position. Therefore, if

two agents attempt to move into the same cell, they both fail and remain in their current place. Note that in stochastic mode, these rules apply to the actual (stochastic) outcome of the action.

Additionally, for each game, we examined the results of learning by heterogeneous agents, that is, agents using different learning algorithms. Finally, to test how well each algorithm scales up with the number of players, we introduced a fourth game in which the state and action spaces do not grow too fast with the number of players. With this game, we were able to play games with up to 5 players.

Adjusting the parameters of the different algorithms was done by a process of trial and error. The algorithms were repeatedly executed in an experimental setup, varying their parameters between executions until some optimum was reached. The parameters that achieved the best performance were then used throughout. Each set of experimental conditions, other than those related to Rmax, was subjected to 100 repeated trials. For Rmax, 20 trials were carried out using $K1 = 50$, 40 with $K1 = 100$ and 40 with $K1 = 200$.[8] The discount factor was 0.98 in all trials. Unless mentioned otherwise, the presented results are averages over all trials.

The following parameter settings were tested:

FriendQ

Exploration: ε-greedy with (i) $\varepsilon_t \leftarrow 1/count_t^{0.5000001}$ where $count_t$ is the number of exploratory steps taken by time t. (ii)$\varepsilon_t \leftarrow 0.99998^{count_t}$. (Unless specified otherwise, (i) is used.)[9]

Learning rate: $\alpha_{s,a} \leftarrow 1/n(s,a)^{0.5000001}$ where $n(s,a)$ is the number of times action a was taken in state s.

Q-value were initialized to 0.

OAL

Exploration: For ε-greedy, $\varepsilon_t \leftarrow 1/count_t^{0.5000001}$, as in FriendQ. For Boltzman exploration, the temperature parameter was decreased by $\tau \leftarrow 100/count^{0.7}$.

History: Random history sample size $k = 5$. History memory size $m = 20$ (m must satisfy $m \geq k \times (n+2)$. OAL with ε-greedy exploration is referred to as ε-OAL, and OAL with Boltzman exploration is referred to as B-OAL.

Q-value were initialized to 0.

Rmax

Sampling: values of 50, 100, 200 and 300 for K1 (visits to mark an entry known) were tested.

Accuracy of Policy Iteration: Offline policy iteration was halted when the difference between two successive approximations was less than 0.001.

8. Because Rmax is a deterministic algorithm, fewer samples were required.
9. Exponential decay of ε violates the infinite exploration condition for convergence.

3.5.1 GAME 1

This game, introduced in Hu and Wellman (1998), was devised to emphasize the effects of equilibrium-selection methods. It has a single goal state (the only reward-yielding state) and several optimal ways of reaching it. The game is depicted in Figure 1. $S(X)$ and $G(X)$ are the respective initial and goal positions of agent X. In the goal state G, both agents are in their goal positions and their reward is 48. Upon reaching the goal, the agents are reset to their initial position. The underlying SG has 71 states. The optimal behavior in deterministic mode reaches G in four steps and yields an average reward per step (a.r.p.s.) of 12.[10] There are 11 different optimal equilibria. In stochastic mode, the optimal policies yield an a.r.p.s. of $\sim$5.285. Algorithms were executed for 10^7 rounds on both settings.

S(A)		G(B)
S(B)		G(A)

Figure 1: Game 1 - initial and goal states.

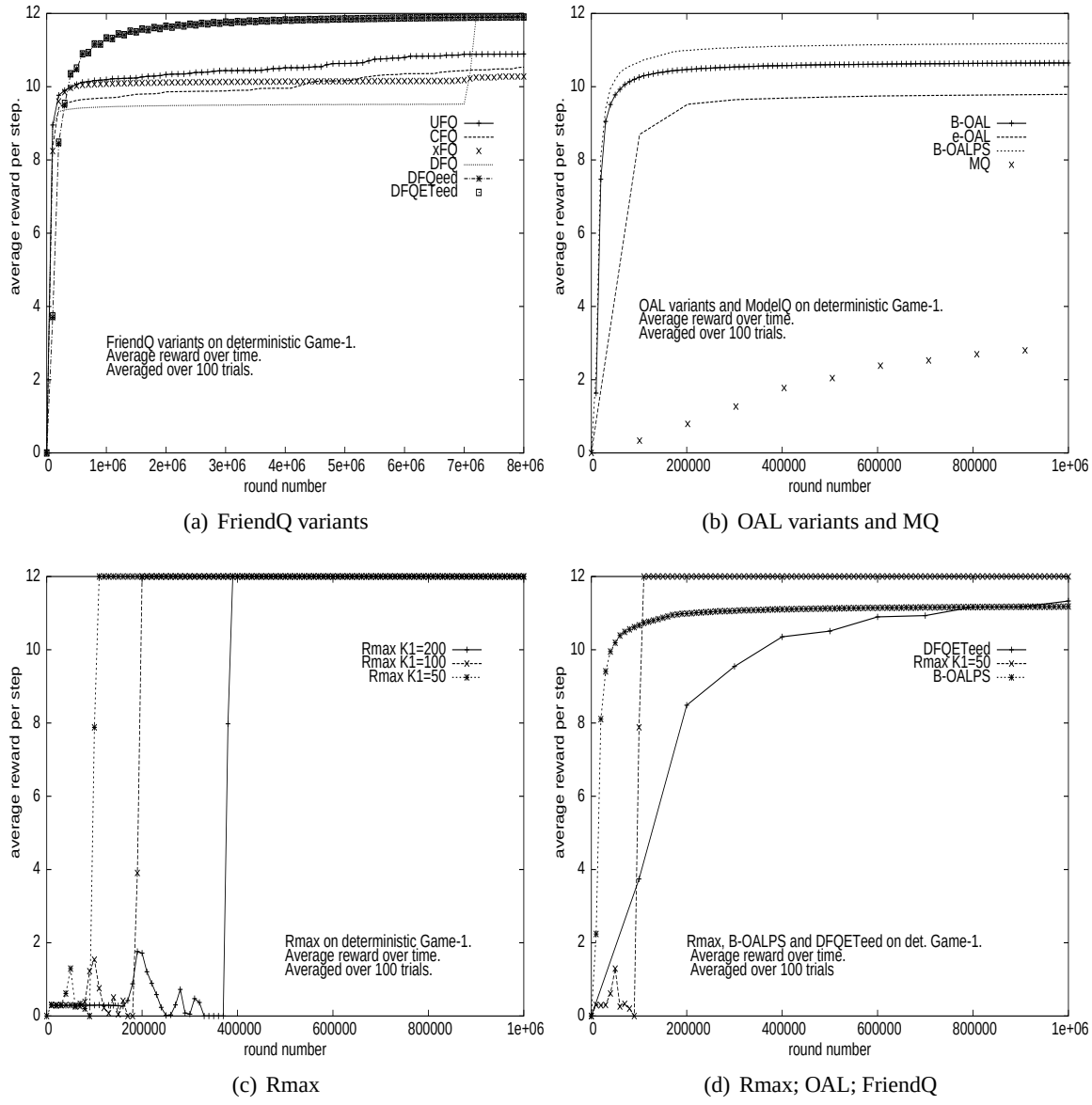
Deterministic Mode

Table 2 reports the number of trials (of 100 in total) in which each algorithm learned a policy, with four levels of final performance based on the number of steps required to reach the goal. For this deterministic domain, we find this measure, which is directly correlated with the more standard a.r.p.s. measure, to be more informative. Here xFQ is a variant of FriendQ in which the agents explore in unison but do not coordinate exploratory actions. The suffix "εed" denotes exponential decay of ε. In the present context, the agents' learning of an optimal policy means that their greedy choice of actions is optimal. That is, with any residual exploration deactivated. Figure 2 presents the a.r.p.s. obtained by the agents over time.

steps to goal	UFQ	CFQ	xFQ	DFQ	DFQεed	ε-OAL	B-OAL	B-OALPS	MQ	Rmax
4	62	49	47	100	100	26	49	41/60	1	100
5	38	49	46			62	51	19/60	49	
6+		2	7			12			29	
∞									21	

Table 2: Game 1 – classification of final performance of learned policies for 100 trials of each algorithm

10. As we noted earlier, our implementation is based on the widely used discounted reward model. But in our goal oriented domains, optimal and near-optimal strategies require a relatively small number of steps to reach the goal. Thus, we chose to report the performance of the learned policies using more intuitive measures such as average reward per step and average steps to reach the goal.



(a) FriendQ variants

(b) OAL variants and MQ

(c) Rmax

(d) Rmax; OAL; FriendQ

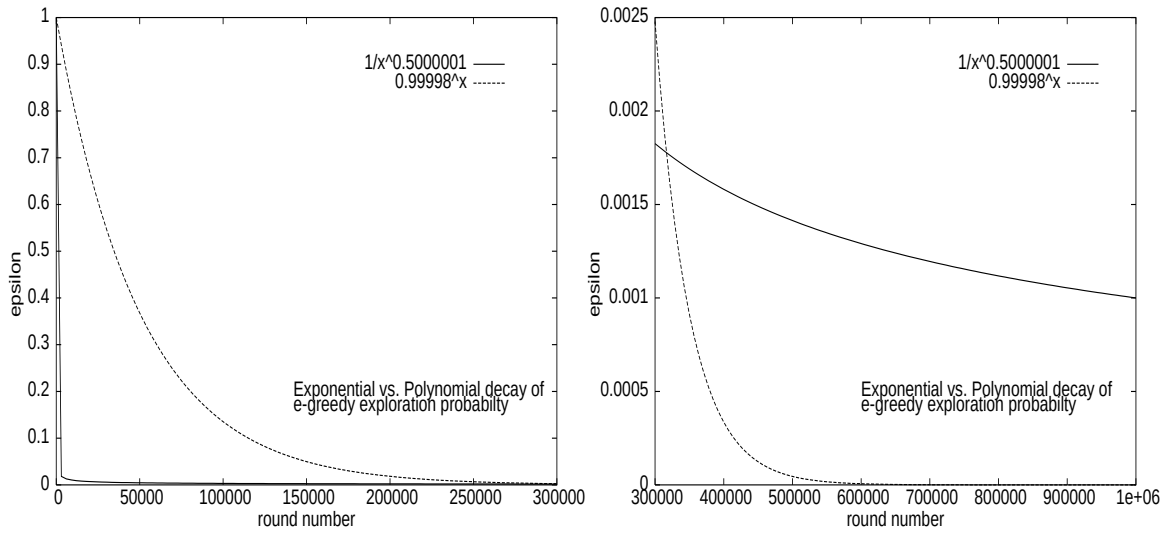
Figure 2: Game 1 – average reward per step under deterministic mode. (a) presents first 8×10^6 rounds. (b), (c) and (d) present first 10^6 rounds.

As can be inferred from the table, in this problem optimal behavior is such that the agents reach the goal in four steps. FriendQ converges quickly to second-best behavior (Figure 2a). From that point on, the average learning curves of UFQ, CFQ and xFQ increase stepwise rather than continuously (although this is a bit difficult to see in the figure). This behavior results from a sudden switch of the FriendQ agents from sub-optimal to optimal behavior once the relative order of the Q-values of different agents changes.

ε-OAL does not present a similar trend. In the trials in which OAL converged to second-best behavior in the first 2.5×10^5 rounds, it failed to find an optimal policy even after 10^7 rounds (Figure 2b). In DFQ, since exploration is deterministic, this switch is always at the same time, specifically after 7×10^6 rounds(Fig. 2a).

Surprisingly, UFQ fares better than CFQ (Table 2, Fig. 2a), in spite of its less sophisticated coordination strategy. At an early learning stage, dis-coordination leads to exploration. Later on, the estimated Q-values of optimal actions are rarely equal, and thus, coordinating exploitation does not pose a problem (at the examined time interval). Exponential decay of ε supplies more exploration at an early period than polynomial decay (Fig. 3) leading to faster convergence of DFQεed (Fig. 2a).

Eligibility traces did not contribute much in this example. The parameters of eligibility traces were hard to tune and very sensitive to change in other parameters or environment dynamics.



(a) Rounds 1 to 300,000

(b) Rounds 300,000 to 1,000,000

Figure 3: Exponential vs. polynomial decay of ε-greedy exploration probability

OAL agents converge relatively quickly to optimal or second-best behavior, and from that time onwards stick to their behavior (Fig. 2b). Whether, in the latter case, they fail to structure the Q-values properly, or the fictitious play prevents the agents from changing their behavior after the Q-values are ordered properly, is not clear from the data. B-OAL converges faster and more often to optimal than ε-OAL (Fig. 2b, Table 2). This behavior seems to stem from the effect of the decay methods we used. The Boltzman method yields more exploration than the ε-greedy method in the early period of learning. Later on, ε-greedy maintains a low exploration probability that decays very slowly while Boltzman exploration drops faster to zero. Thus, even when ε-OAL learns optimal behavior, it keeps achieving only near-optimal average-reward.

As expected B-OALPS improves on the performance of B-OAL (Fig. 2b, Table 2) because of its more rapid propagation of learned information.

The performance of ModelQ is inferior to that of OAL (Fig. 2b, Table 2), presumably because ModelQ does not explore as much as OAL: At early stages of learning, fictitious play provides OAL with other means of exploration. When the agents make many stochastic action choices in

early stages of learning, fictitious play amplifies the random behavior. However, at later stages of learning, deviation from constant action choice is rare and will probably not affect fictitious play. In this setting, ModelQ shows slower convergence than the model-free FriendQ.

The learning graph of Rmax can be precisely divided into two periods, an initial learning period in which Rmax attains very low return due to exploration, followed by a period of exploitation in which Rmax attains optimal return (Fig. 2c). The length of the initial period depends linearly on $K1$.[11] Figure 2d compares the best performing variant of each algorithm.

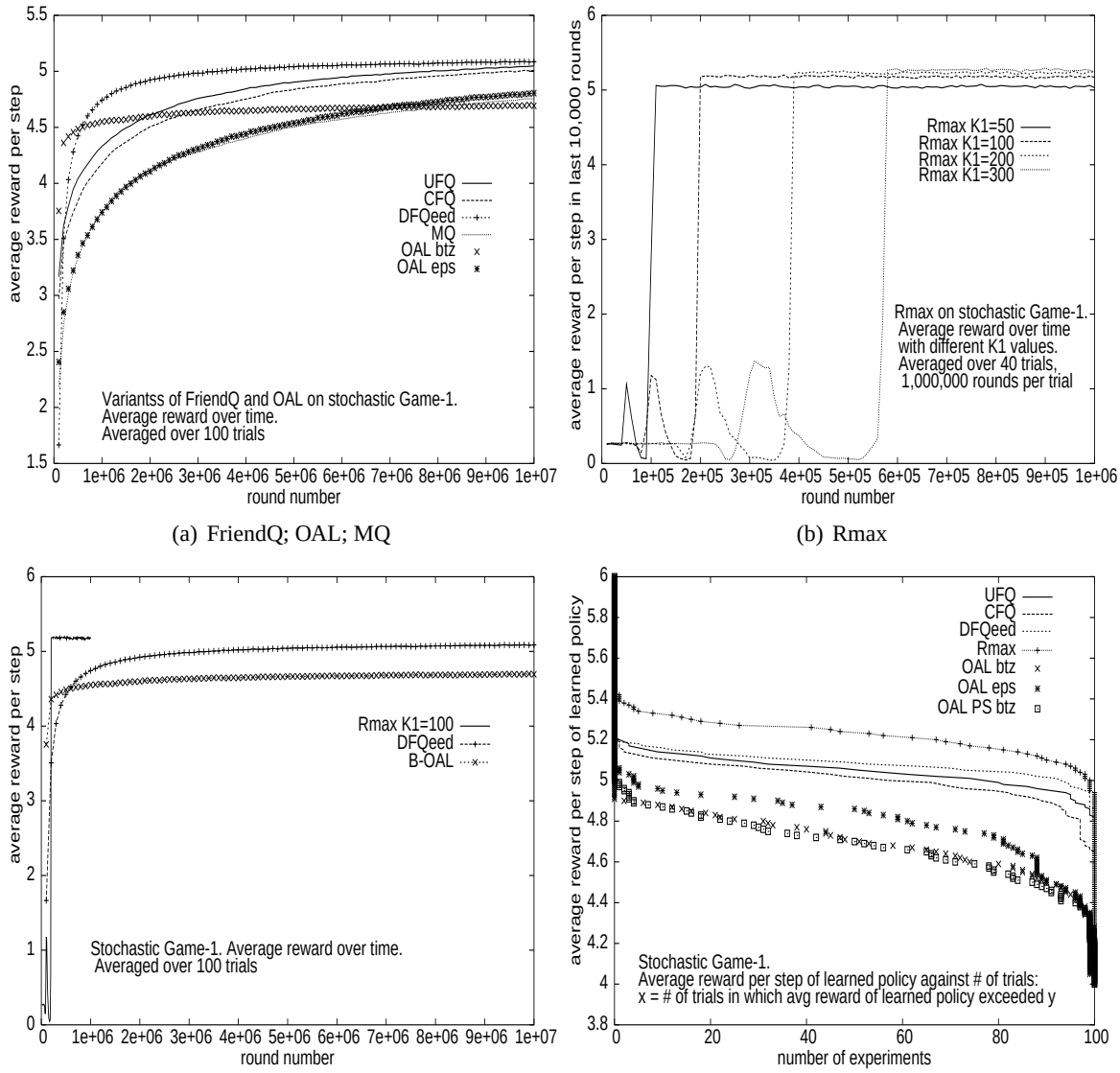
Stochastic Mode

Figure 4 presents the results for the stochastic mode. As expected, due to the stochastic effects of actions, the value of the optimal policy decreases, and more importantly, the learning algorithms require more trials to converge. By contrast to the deterministic case, MQ performs as well as ε-OAL (Fig. 4a). This improvement is attributable to additional exploration stemming from the stochastic nature of the environment. For the same reason, CFQ performs almost the same as UFQ (Fig. 4a). When we compare the gap between the DFQεed to U/CFQ at the first 10^6 rounds in stochastic mode vs. the deterministic mode we find that the gap is smaller. This difference is due to the fact that the additional early exploration supplied by the exponential decay of ε is redundant in the stochastic case. The slightly higher return gained by DFQεed later on is due to the faster decay of ε. Another interesting difference from the deterministic setting is that initially ε-OAL gains lower return than B-OAL but while B-OAL keeps attaining the same average reward, ε-OAL improves slowly over time and eventually gains a higher average reward than B-OAL. In this case, the slower convergence of the exploration probability to zero enables ε-OAL to "overcome" randomly "bad" exploration in initial learning phases.

Rmax behaves similarly in stochastic and deterministic modes. While the other algorithms achieve only near-optimal return, Rmax attains optimal return (Fig. 4b,c). Rmax's strong exploration bias results in low return until model entries are known. From that point on, Rmax attains an optimal return. The histogram (Fig. 4d) shows that Rmax converges to higher return than the other algorithms not only in the average case but also in the worst case (i.e., almost all runs of Rmax were better than the best runs for the other algorithms). Very low values for $K1$, which mean rough transition probability estimates, are enough for computing near-optimal behavior. Indeed, the exploration vs. exploitation tradeoff is evident even in this simple example. We see how a smaller value of $K1$ leads to faster convergence, but at the cost of slightly smaller average reward.

Overall, it appears that the major issue for the FQ and OAL class of algorithms is exploration. As the space of joint-actions is quite large, there are many relevant options to try. Especially in the deterministic case, the rather standard exploration techniques we used appear to be insufficient. Although stochastic domains naturally lead to more exploration, we can see that the model-free algorithms are sub-optimal. It appears that model-free algorithms—at least in their standard form—have difficulty determining whether certain states were explored sufficiently, and that standard exploration schemes are too crude. Overall, many of the phenomena observed in Game 1 were present in Games 2 and 3. Therefore, in the following experiments, only phenomena not observed in Game 1 will be emphasized.

11. If it is known ahead of time that the environment is deterministic, then $K1$ can be set to 1. Similar locality considerations on stochastic environments can help determine tight bounds on $K1$.



(a) FriendQ; OAL; MQ

(b) Rmax

(c) Rmax; FriendQ; OAL

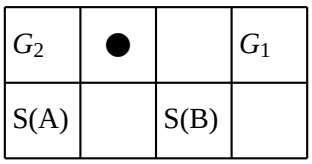
(d) Average reward of learned policies per number of trials

Figure 4: Game 1 – Average reward per step and learned policies per number of trials in stochastic mode. Subfigures (a) and (c) present all 10^7 rounds, while (b) presents the first 10^6 rounds.

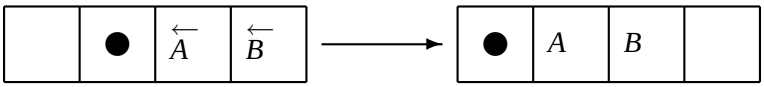
3.5.2 GAME 2

This game was designed to minimize the effects of equilibrium selection, to show how GLIELPs may keep agents exploiting suboptimal possibilities, and to emphasize the importance of coordinated exploration. The game has four goal states and one optimal equilibrium. The game is depicted in Figure 5(a). It consists of an additional element, an object that can be moved by the agents. The agents can move in four directions or stay in place. They can push the object by standing to its right(left) and moving left(right) and pull the object by standing to its right(left) and

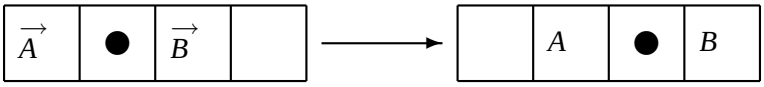
moving right(left). However, the object is too heavy for one agent and requires cooperation of the two agents to be moved. The manner by which the object is moved is depicted in Figure 5(b). Note that the push/pull effect is a by-product of the agents' moves. Thus, in stochastic mode, what determines if the action is push or pull is not the chosen action but its actual effect.



(a) Initial state and Goal states.



(b1) Moving the object by pushing simultaneously.
(Agents' order does not matter).



(b2) Moving the object by pushing and pulling simultaneously.
(Agents' order does not matter).

Figure 5: Game 2

The agents' goal is to move the object into one of the upper corners of the grid, at which point the game is reset to its initial state. Moving the object to the upper right (G_1) or left (G_2) corner yields a reward of 80 and 27, respectively. The optimal behavior under deterministic mode is to move the object to G_1 in 8 steps. The average reward per step of an optimal strategy under deterministic mode is 10, and the discounted return is ~ 465. The second-best strategy is moving the object to G_2 in 4 steps, with an a.r.p.s. of 9 and discounted return of ~ 440. In stochastic mode, the optimal policy may stochastically lead to one of the goal positions. The a.r.p.s. of the optimal policy in stochastic mode is ~ 3.8. The underlying CISG contains 164 states. Algorithms were executed for 3×10^7 rounds.

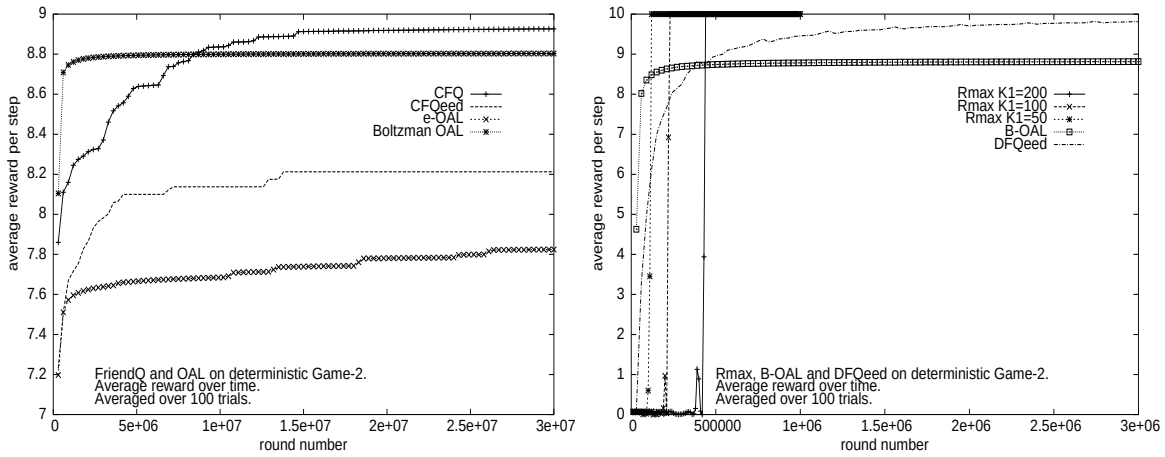
Deterministic Mode

Table 3 classifies the number of trials (of 100 per algorithm) according to the algorithms and learned policies. Figure 6 shows the a.r.p.s. over time obtained by the different algorithms.

The main reasons for the sub-optimal performance of OAL and CFQ in this game are: (i) Random exploration has a greater chance of reaching G_2 than G_1. Discovering G_2 before G_1 further reduces the chance of visiting G_1 because of the increasing bias toward exploitation. (ii) Exploration of the CFQ and OAL agents is not coordinated. If reaching G_2 is the current greedy policy, then G_1 will not be visited unless both agents explore simultaneously. Game 1 demonstrated an advantage of exponential decay of the ε-greedy exploration probability over polynomial decay of this probability. Game 2 demonstrates an opposite phenomenon, Fig. 6a and Table 3 show that CFQ does better with polynomial decay of ε than with exponential decay. This result stems from finite exploration supplied by exponential decay. However, this finite amount of exploration is sufficient

Goal	steps to goal	CFQ	CFQεed	DFQεed	ε-OAL	B-OAL	Rmax
G_1	8			100		1	100
G_2	3	99	65		54	91	
G_2	4	1	35		46	8	

Table 3: Game 2 – Characteristics of the learned policy on a per-trial basis for each algorithm in deterministic mode.



(a) FriendQ and OAL.

(b) Rmax; B-OAL; DFQeed.

Figure 6: Game 2 – Average Reward under deterministic mode. Subfigure (a) presents all 3×10^7 rounds; Subfigure (b) presents first 3×10^6 rounds.

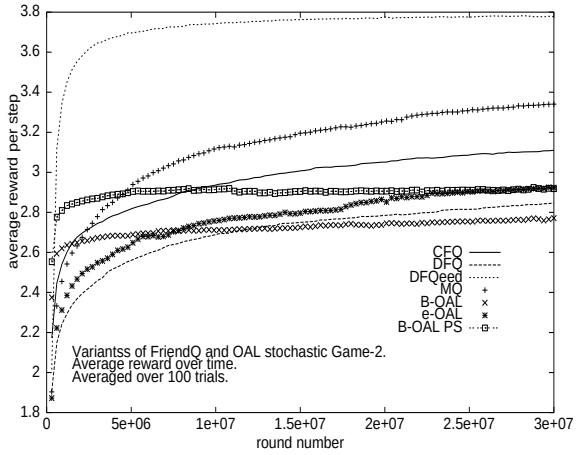
when exploration is coordinated as shown by the learning curve of DFQεed (Fig. 6b) and by Table 3. Furthermore, DFQεed converges to optimal greedy behavior while both CFQ variants do not.

Stochastic Mode

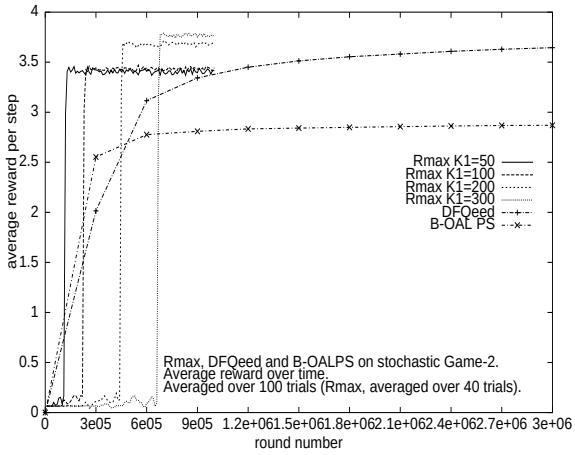
Figure 7 presents statistics for the stochastic mode. It exhibits two interesting phenomena that have not been observed in the previous experiments. One is that, in contrast to previous results, ModelQ outperforms ε-OAL (Fig. 7a,c). Since the only difference between ε-OAL and ModelQ is the greedy action selection method, a reasonable explanation is that BAP (OAL's action selection mechanism) delays behavioral change that should follow Q-value updates (which in turn may delay learning of Q-values). This outcome is because BAP plays a best response to the strategy implied by the other agent's past plays. Since both agents react to each other's past plays using BAP, it may take long to converge to a new NE when the optimal joint actions are changed. The second phenomenon is that Rmax requires larger values of $K1$ to converge to optimal behavior (Fig. 7b). This finding can be explained by the fact that the optimal behavior involves longer cycles of state transitions and hence the model has to be more accurate.

As in Game 1, we see that exploration strategies have a great impact on the ability of different algorithms to converge. In this respect, Game 2 highlights the need for coordinated exploration.

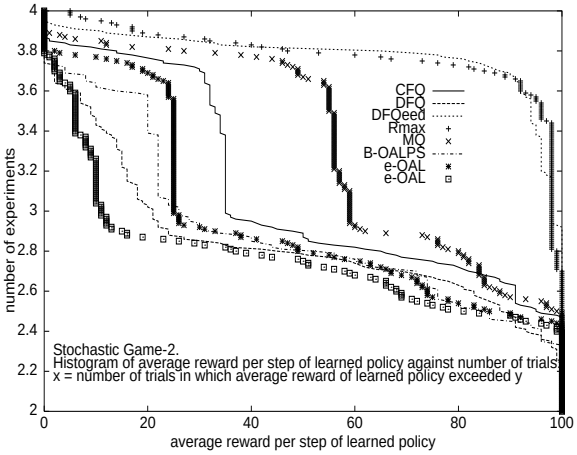
Thus, in cooperative multi-agent systems, we face the standard problem, clearly visible in Game 1, of ensuring sufficient exploration, but we need to ensure that this exploration is effective by coordinating exploratory moves of different agents.



(a) FriendQ and OAL



(b) Rmax; B-OALPS; DFQeed



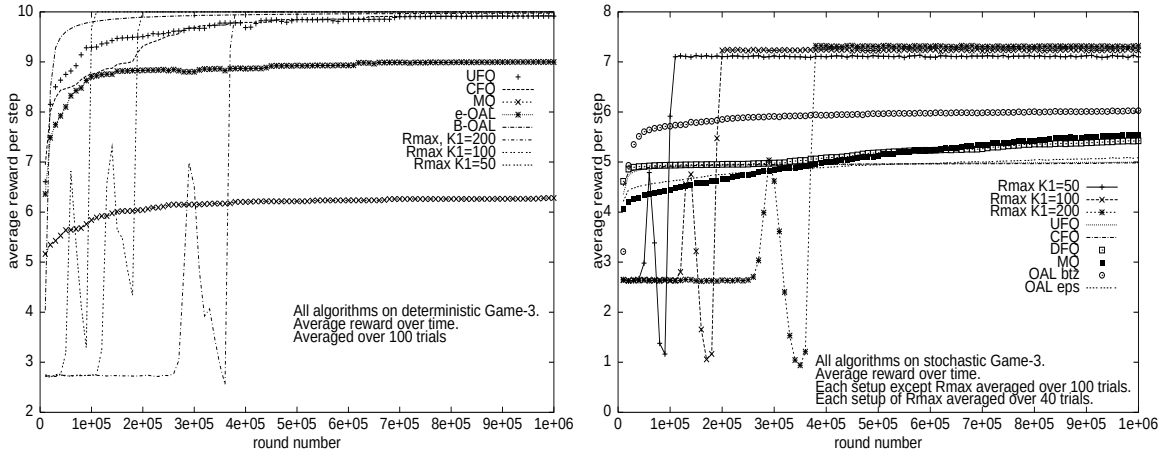
(c) Average Reward of Learned Policies per Number of Trials

Figure 7: Game 2 – Average reward per step and learned policies per number of trials under stochastic mode. (a) presents all 3×10^7 rounds; (b) presents first 3×10^6 rounds.

3.5.3 GAME 3

In the previous games, one had to explore considerable parts of the state space in order to construct good policies. This game is characterized by a maximum return attainable by staying in a small local set of states anywhere on the state graph. The initial position of the agents within a 3×3 grid is random. They are rewarded for reaching a position in which their locations are adjacent. If this position is attained by unaltered positions of both agents, the reward is 5. If movement is involved, the reward is 10. Algorithms were executed for 10^6 rounds.

As opposed to previous experiments, in deterministic mode, B-OAL and the FriendQ variants converged faster to optimal (greedy) behavior than Rmax (Fig. 8a). Rmax explores the whole model before it starts exploiting while FriendQ's and OAL's choice of greedy actions is optimal long before good estimates of all Q-values are attained. However, in stochastic mode the GLIELPs no longer have this advantage since stochastic transitions do not enable the agents to concentrate on exploiting a local set of states (Fig. 8b).



(a) deterministic mode (b) stochastic mode

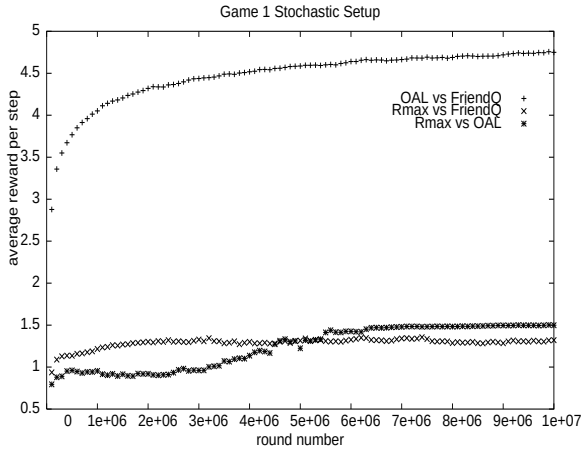
Figure 8: Game 3 – average reward per step under deterministic and stochastic modes.

3.5.4 HETEROGENEOUS PLAYERS

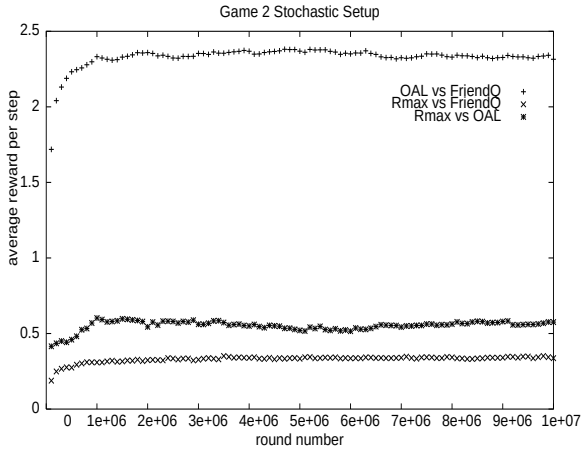
A CISG is most naturally viewed as a model of a distributed stochastic system. As such, it is natural to have in mind a view of a system's designer, and one would expect such a designer to equip the players with identical algorithms. However, CISGs arise also when self-interested agents need to coordinate, typically on the use of some resource, where coordination is beneficial to all parties involved. Examples include which side of the road to travel on, the meaning attached to a symbol, etc. Thus, it is natural to ask how the algorithms tested fare in the context of other algorithms. We reran the above experiments with pairs of different algorithms. The results, presented in Figure 9 are, quite uniform (similar performance is observed in the deterministic games). The top performance, and as is clearly visible, by a wide margin, was obtained by OAL+FriendQ. Pairs containing Rmax performed much worse, with Rmax+OAL typically fairing slightly better than Rmax+FriendQ. In fact, comparing the results to the homogeneous case, the OAL+FriendQ combination performed almost optimally in Game 1: 4.7 vs. 5. In Game 2 it obtained 2.3 vs. 3.8, and in Game 3 it achieved 5.85 vs. 7.2.[12] And while Rmax and, to a lesser extent, FriendQ do better against their own kind, OAL does better against FriendQ. It may be the case that for equilibrium selection, OAL's mechanism works best when one agent "insists" more on a particular equilibrium, thus more quickly breaking up symmetries.

These results might be interpreted as an indication of the "rigidity" of each of the algorithms. FriendQ is the simplest of the three algorithms, it makes no internal assumptions about its partners

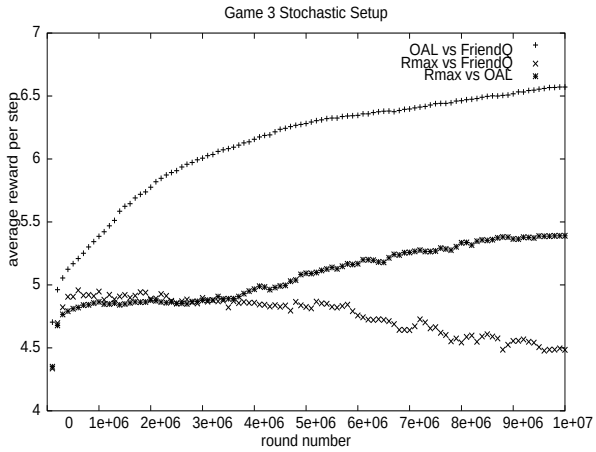
12. The version of FriendQ used was CFQ with replacing traces. The OAL version used ε-greedy exploration.



(a) Game 1 stochastic mode



(b) Game 2 stochastic mode



(c) Game 3 stochastic mode

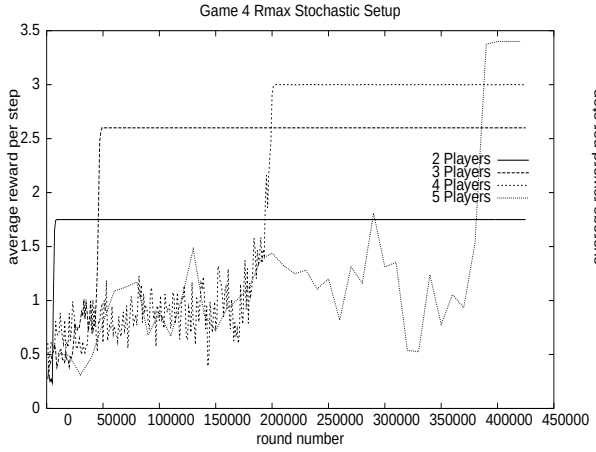
Figure 9: Heterogeneous play games 1-3.

and simply adapts. Rmax is at the other extreme, it strongly relies on the behavior of its partners in order to systematically explore and then exploit. OAL is somewhere in between. It does have a sophisticated mechanism for selecting among different equilibria, but this mechanism is stochastic and can handle noise, and is based on fictitious play, which is a mechanism that adapts to the empirical behavior of the other agents. Thus, one would expect Rmax to fail when its assumptions are not met, as its implicit coordination mechanism is based upon them. In contrast, FriendQ and OAL, which make weak internal assumptions about their peers, should work well, especially when their opponent shows some flexibility and adaptivity.

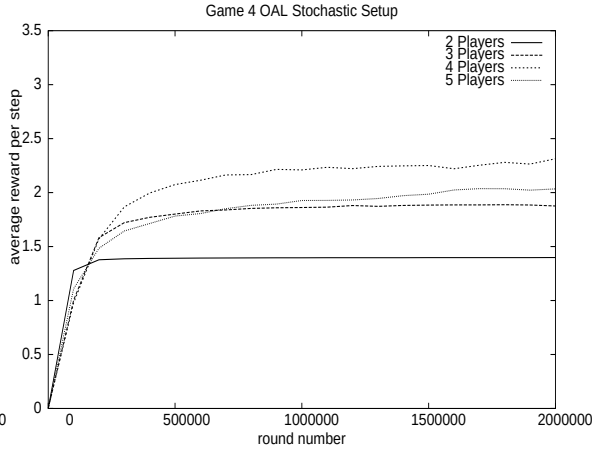
3.5.5 $n > 2$ PLAYERS

So far, we considered only two-player games. The reason is practical: Experiments conducted on large state spaces take long to execute. It is especially true for Rmax which must solve the MDP each time the model changes. This effort grows with the state-space, and the state-space grows

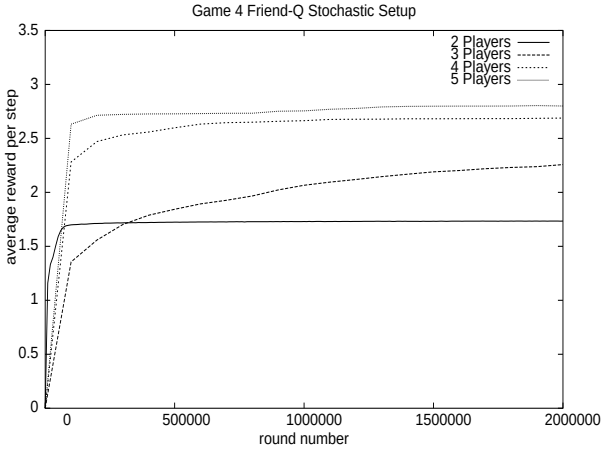
exponentially with the number of players. Thus, to get an idea of how these algorithms fare with a large number of players, we devised a simpler, fourth game in which we could run experiment with up to 5 players. This is a simple linear grid with 5 positions. Players can move to the left and the right. When two players attempt to move to the same position, the result is with probability 1/3 none move, and with probability 1/3 each one of the players makes the move and the other stays in place. In the initial state, player i is in position $5-i$. The goal position of each player is i. Generally, the reward at each state is the number of players located at their goal positions. However, when all players are in their goal position, they receive a reward of 3 *times* the number of players, at which point all players transition automatically to the initial position.



(a) Game 4 R-max



(b) Game 4 OAL



(c) Game 4 Friend-Q

Figure 10: Results for 2-5 players on game 4.

The results are presented in Figure 10. Note the difference in scale for the X-axis for OAL and FriendQ, which is intended to show that the suboptimal a.r.p.s. to which they converge does not increase even when we look at millions of steps. As in the previous section, we used the CFQ version of FriendQ with replacing traces, ε-greedy exploration for OAL, and $K_1 = 100$ for Rmax.

For all algorithms, the value is greater as the number of players increases due to the game's reward definition, which is sensitive to the number of players. All algorithms converge quickly on this game for all number of players. However, the values they converge to differ. Among the three algorithms, Rmax converges to the highest average per step reward, FriendQ is next, and OAL is last. The relative performance is consistent with the performance displayed in the three earlier two-player games. This finding is a reasonable indication that the relative performance of these algorithms is qualitatively similar regardless of the number of players, at least for small player sets.

3.5.6 SUMMARY

Section 3.5 presents an experimental study of three fundamentally different algorithm families for learning in CISGs. The results illustrate the strengths and weaknesses of different aspects of these algorithms in different settings, highlighting the accentuated importance of effective exploration, which is enabled in this class of games only by coordinated behavior, the advantage of deterministic behavior for attaining such coordinated behavior, and the benefits of propagation of information.

Each of the experimental domains emphasizes different aspects of the learning task in CISGs. The results show that the parameters of OAL and FriendQ are very sensitive to environmental topology and dynamic. Exploration and coordination strategies suitable for one environment may be very inefficient in other environments. Rmax, on the other hand, is stable in this respect. It has a single parameter, $K1$, that has to be preset. Its convergence time depends linearly on $K1$ (and the size of the state-action space) and it turns out that Rmax converges to near optimal behavior using values of $K1$ that achieve faster convergence than that of OAL and FriendQ. However, the convergence dynamics of Rmax does not suit tasks in which the agents must attain some value during the learning period, because during its exploration phase, Rmax is indifferent to rewards lower than R_{max}. However, Rmax is also the simplest algorithm, and thus it is easy to alter it, for instance, to obtain satisficing behavior, for example, by lowering the value of R_{max} in the model, or by starting with a moderate value and then increasing it as better values are observed. Overall, when we control the algorithm of all agents in the system, Rmax seems to be the best alternative—it converges quickly to values higher than those of OAL and FriendQ, it does not seem to be sensitive to an increase in the number of players, except through its effect on the state space, and most importantly, it has very simple exploration strategy. As we saw in games 1 and 2, the choice of exploration strategy has much influence on the results of FriendQ and OAL, and the precise choice is sensitive to the nature of the game, number of equilibria and their nature. Thus, the simple exploration behavior of Rmax and the potential to alter it in various transparent ways is a clear benefit. Yet, when we may need to coordinate with other players with unknown coordination mechanisms, or if our underlying state space is too big for repeated value computations, FriendQ seems to offer the best choice.

4. Learning in FSSGs

In two-player *Zero Sum* SGs (ZSSGs), the players' payoffs sum up to zero at every entry. That is, $[R(s,a^1,a^2)]_1 = -[R(s,a^1,a^2)]_2$ for every $s \in S$, $a^1 \in A_1$ and $a^2 \in A_2$. Such payoffs indicate that the agents' interests completely conflict. A ZSSG can be modeled with a single payoff function $R'(s,a^1,a^2) = [R(s,a^1,a^2)]_1$ by redefining Player's 2 objective as to minimize the IHDR (infinite horizon discounted reward). For the rest of this section, it is assumed that Player 1 is the maximizer and Player 2 is the minimizer the payoff function R. Let $V(s,\pi_1,\pi_2)$ denote the expected IHDR for starting at state s and playing the profile (π_1,π_2) of stationary mixed policies there-

after, and $V(\pi_1,\pi_2) = (V(1,\pi_1,\pi_2),...,V(|S|,\pi_1,\pi_2))$. Since the best response in a ZSSG is also the worst for the opponent, ZSSGs have a unique NE value. To see this, assume by negation that $V(s,\pi_1,\pi_2) > V(s,\mu_1,\mu_2)$ and that both (π_1,π_2) and (μ_1,μ_2) are NE. Since $\pi_2 \in BR(\pi_1)$, it follows that $V(s,\pi_1,\mu_2) \geq V(s,\pi_1,\pi_2) > V(s,\mu_1,\mu_2)$, which contradicts $\mu_1 \in BR(\mu_2)$. The value of a policy π may be defined as $V(\pi,BR(\pi))$. In ZSSGs, this definition coincides with that of a NE (any pair of optimal policies is a NE and vice versa).

Consequently, the state-value function $V(s)$ is redefined to be the expected IHDR under a profile of optimal policies and $Q(s,a_1,a_2)$ the expected IHDR for taking joint action (a_1,a_2) in state s and continuing according to a NE thereafter. For any stationary strategy profile (π_1,π_2) in a ZSSG G, $(\pi_1(s),\pi_2(s))$ is a NE for the matrix games defined by $[Q(s,a^1,a^2)]_{a^1\in A_1,a^2\in A_2}$ for all $s \in S$ if and only if (π_1,π_2) is a NE for G and the NE values for the matrix games correspond to the state values $V(s,\pi_1,\pi_2)$ (Filar and Vrieze, 1997). Thus, the Bellman optimality equations can be rewritten for ZSSGs as

$$\begin{aligned} Q^*(s,a^1,a^2) &= R(s,a^1,a^2) + \gamma \sum_{s'} T(s,a^1,a^2,s')V^*(s') \\ V^*(s) &= \sum_{a^1\in A_1,a^2\in A_2} \pi_1(a^1)\pi_2(a^2)Q(s,a^1,a^2) \end{aligned} \quad (1)$$

where (π_1,π_2) is a NE for the matrix game defined by the Q-values in state s. Given a method that computes NE for zero sum matrix games, Equation (1) can be used as an iterative approximation rule to compute the Q-values (Littman, 1994) and given the Q-values an optimal policy can be derived. The NE policies for a zero sum matrix game $M = [r(a_i,b_j)]_{i=1,j=1}^{k,l}$ are the solutions to the linear program that maximizes v under the constraints (Filar and Vrieze, 1997)

$$\left\{ \sum_{i=1}^{k} \pi(a_i) r(a_i,b_j) \geq v \mid j \in \{1,...,l\} \right\}.$$

In the following sections, this linear program is abbreviated as:

$$v = \max_{\pi\in PD(A)} \min_{b\in B} \sum_{a\in A} \pi(a) r(a,b).$$

If SG G is obtained from ZSSG G' by adding a constant c to all payoffs of **both** players, then $V_i^G(\pi_1,\pi_2) = V_i^{G'}(\pi_1,\pi_2) + c/(1-\gamma)$ for any policy profile (π_1,π_2) and the strategic properties of the game are unchanged. G is referred to as a Fixed Sum Stochastic Game (FSSG). The adversarial nature of FSSGs calls for agents that perform well not only in self play but also in *heterogeneous play*, namely when engaged by agents that employ different learning algorithms. Under this setting, the exploration/exploitation tradeoff wears a new guise as attempted exploration and exploitation may be interfered by the opponent.

This section compares three algorithms for learning in FSSGs: FoeQ (Littman, 1994, 2001), WoLF (Bowling and Veloso, 2002) and Rmax (Brafman and Tennenholtz, 2002). They were selected because they represent different approaches to the exploration/exploitation tradeoff and to information propagation while providing some theoretical guarantees. Specifically, Rmax and FoeQ converge to a NE in FSSGs in self-play, while WoLF is known to converge in 2 player, 2 action, repeated games.

4.1 FoeQ

FoeQ (a.k.a MinimaxQ) (Littman, 1994, 2001) extends Q-learning into FSSGs by using a sample backup learning rule based on Equation 1 (Littman, 1994). After taking a joint action (a,b) in state s at time t and reaching state s' with reward r_{cur}, the agent updates the Q-value of $\langle s,(a,b)\rangle$ by

$$Q_t(s,a,b) \leftarrow (1-\alpha_t)Q_{t-1}(s,a,b) + \alpha_t \left(r_{cur} + \gamma \max_{\pi \in PD(A)} \min_{b' \in B} \sum_{a' \in A} \pi(a') Q(s',a',b') \right).$$

Q_t converges in the limit to Q^* under the standard Q-learning conditions stated in Section 3.1. Also, for similar reasons to those stated in Section 3.1, FoeQ is executed with an ε-greedy learning policy.

4.2 WoLF

WoLF (Bowling and Veloso, 2002) is designed to converge to a best response rather than a NE. WoLF does not explicitly consider an adversary. It applies the standard single-agent Q-learning rule to approximate Q-values of private actions and uses hill climbing to update its mixed policy. That is, the policy is improved by increasing the probability of selecting a greedy action according to a policy learning rate δ (which is distinct from the Q-value learning rate α), enabling mixed policies. The uniqueness of WoLF is in using a *variable* policy learning rate according to the "Win or Learn Fast" (hence WoLF) principle: if the expected return of the current policy given the current Q-values is below (above) a certain threshold then a high, δ_l (low, δ_w), learning rate is set. A good threshold would be the NE value of the game because if the player is receiving less than its value, its likely playing a sub-optimal strategy, whereas if it receives more than the NE value, the other players must be playing sub-optimally. Since the NE value is unknown, it is approximated by the expected return of the average policy (averaged over the history of the game) given the current Q-values. The motivation for the WoLF variable policy learning rate is to enable convergence to a NE. Indeed, Bowling and Veloso (2002) show that gradient ascent with WoLF is guaranteed to converge to a NE in self play on two-player, two-action, repeated matrix-games, while gradient ascent without a variable learning rate is shown not to. Furthermore, they provide empirical results on FSSGs in which WoLF converges to NE in self play. WoLF, as single agent Q-learning, is guaranteed to converge in the limit to a best response under the standard conditions and given that the opponent(s) converge to stationary policies.

4.3 Rmax

Section 3.3 describes the Rmax algorithm in the context of MDPs. The same algorithm is applicable to FSSGs with the only difference that joint actions are considered and optimal policies with respect to the fictitious model are computed according to (1). As mentioned in Section 3.3, Rmax always behaves optimally with respect to an approximated, initially optimistic, model M' of the real model M. Since unknown entries are modeled in an attractive manner in M', Rmax has a strong bias to explore. Seeing that the optimal policy maximizes return against the worst opponent, if the opponent prevents Rmax from visiting unknown entries then Rmax attains near-optimal return because the known entries are accurately modeled. Thus, Rmax is guaranteed to either attain near optimal return in the real model M or, with sufficiently high probability, visit unknown entries (Brafman and Tennenholtz, 2002). This property assures that Rmax will attain near-optimal average reward after a polynomial number of steps in FSSGs as well as in MDPs.

As in the CISGs experiments, $K1$, the number of visits required to declare an entry known, is treated as a parameter that has to be preset and R_{max} is not assumed to be known. Instead, R_{max} is initialized to some positive value and updated online to be twice the highest reward encountered so far.

4.4 Discussion of Algorithms

The shortcoming of GLIELPs discussed in Section 3.4, namely the possibility of untimely greediness, applies also to FoeQ and WoLF in fixed sum environments and may be further exploited by an informed adversary. Furthermore, FoeQ and WoLF do not reason about how the opponent affects exploration. Thus, attempted exploration may result, depending on the opponent's action choice, in joint actions that are of low informative and materialistic value. FoeQ's and WoLF's single step backups and possible premature decrease of the Q-learning rates may cause poor use of new experience.

WoLF compensates for the above limitations by the following properties: (i) Hill climbing adjustment of the policy for enhanced exploration. Specifically, this exploration is regulated by the variable policy learning rate to explore more while "winning". The gradual policy update also prevents formation of big gaps between Q-values of different entries and thus contributes to both fast adjustment to changes in the adversary's behavior and reduction of the effect of untimely greediness. (ii) WoLF's greedy policy is a best response rather than a NE. This fact results in high payoffs during learning, fast growth of Q-values and hence fast convergence. (iii) WoLF does not explicitly model the opponent. It maintains Q-values for the small action space of private actions resulting in faster propagation of state-action values.

A major conceptual difference between WoLF and both Rmax and FoeQ is the target of learning, which also implies the definition of greediness during learning. Rmax's and FoeQ's greedy policies are NE policies. Playing a NE policy is the best strategy against a rational opponent. It also makes sense even if the adversary does not play a BR since it ensures at least the value of the game. However, playing a NE policy w.r.t. a non-accurate model/Q-values during learning makes the hidden assumption that the opponent is not only rational but also acts according to the same model/Q-values. Under heterogeneous play, this assumption is not valid and may result in low payoffs during learning. FoeQ typically maintains pessimistic Q-values during learning. The resulting (greedy) NE learning policy will attempt to avoid entries that are not well known since they seem unprofitable. Thus, FoeQ's greedy action choice may have low informative and materialistic value when engaged in heterogeneous play. For model free algorithms, fast convergence depends on high payoffs during learning. Rmax does not distinguish exploration from exploitation and guarantees to either exploit or explore independent of the adversary's actions. Thus, low payoffs during learning are traded for faster convergence to a NE policy. However, Rmax is biased to explore and may play exploratory actions even when it "knows" a submodel in which the value is attainable. WoLF pursues the best response policy during learning. This strategy is efficient against adversaries that converge to stationary policies. However, an adversary that knows WoLF's strategy may play a "decoy" policy until WoLF's learning is slow and then switch to a best response.

Notwithstanding formal results (Even-Dar and Mansour, 2003), parameter tuning is still a task that requires expert experience and intuition. In this respect, WoLF is the most complicated among the three algorithms. On top of the parameters for decaying exploration and Q learning rates, which also appear in FoeQ, it involves presetting the decay rate of the policy-learning rates and the relation

between the two policy-learning rates δ_w and δ_l. Rmax is the simplest to tune among the three with a single parameter, $K1$, the number of visits to declare an entry "known".

Finally, We note that our discussion of convergence rates and the especially our experimental evaluation focuses on the number of time steps,or multi-agent encounters rather than CPU-time. Although the algorithms we evaluated are all considered practical for online reinforcement learning, it should be noted that WoLF requires considerably less computation than FoeQ or Rmax since it does not involve linear programming computations of equilibria.

4.5 Experimental Results & Analysis

This section describes experimental results on three 2-agent fixed-sum grid games. The games were designed to evaluate the effects of exploration, information propagation, action selection and other methods, on the performance of FoeQ, WoLF and Rmax in different environments. The algorithms were tested in both self play and heterogeneous play. The available actions, indexing of the grid, transition probabilities for the deterministic and stochastic modes, discount factor etcȧre the same as in the CISG experiments. The process of adjusting the parameters was also similar to the CISG experiments and was conducted on the "deterministic 3×3 Wall game" (see below).

The following parameter settings were used for testing:

FoeQ

Exploration: ε-greedy, $\varepsilon_t \leftarrow \max\left\{0.99999^{count_t}, 1/count_t^{0.5000001}\right\}$ where $count_t$ is the number of exploratory steps taken by time t.

Learning rate: $\alpha_{s,a} \leftarrow \max\left\{0.99908^{n(s,a)}, 1/n(s,a)^{0.75}\right\}$ where $n(s,a)$ is the number of times action a was taken in state s.

Q-values were initialized to 0.

WoLF

Exploration: ε-greedy, $\varepsilon_t \leftarrow \max\left\{0.5000001^{count_t}, 1/count_t^{0.5000001}\right\}$.

Q Learning rate: $\alpha_{s,a} \leftarrow \max\left\{0.95^{n(s,a)}, 1/n(s,a)^{0.0.5000001}\right\}$.

Policy Learning rate: $\delta_l = 0.7 \times \alpha_{s,a_g}, \quad \delta_w = 0.175 \times \alpha_{s,a_g}$ where a_g is a greedy action in the current state s.

Rmax

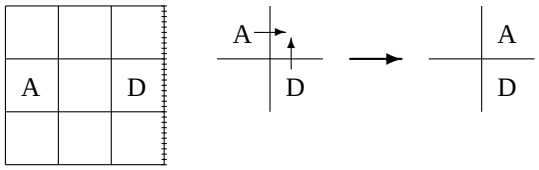
Sampling: values of 50, 100 and 200 for $K1$ (visits to mark an entry known) were tested.

Accuracy of Policy Iteration: Offline Policy Iteration was halted when the difference between two successive approximations was less than 0.001

4.5.1 3×3 WALL GAME

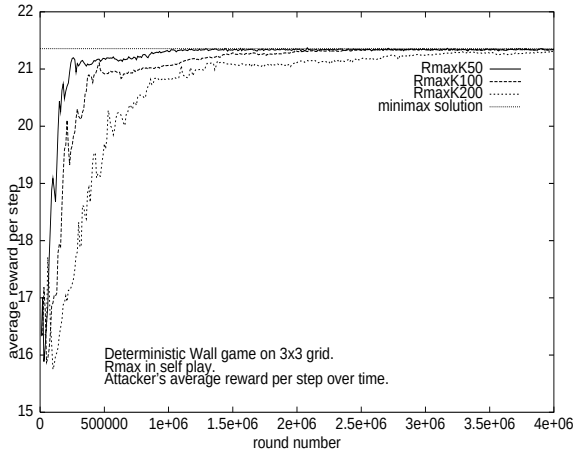
In this 3×3 grid game, one player, A, is an Attacker and the other, D, is a Defender. Figure 11a depicts the initial position of the game. A's goal is to reach the rightmost column of the grid. If both players try to enter the same square or to enter each other's current positions (that is, switch places) then their locations are unchanged. The only exception to this rule is when the players are in diagonally adjacent squares—in this case A moves and D's position is unchanged (Fig. 11b), so that

the attacker has a slight advantage. The fixed sum of the game is 40. When A reaches the rightmost column of the grid, it receives a reward of 40, D receives a reward of 0 and the players are reset in their initial positions. For any other move, A is rewarded by 15 and D is rewarded by 25. The game was played under deterministic transition probabilities. Every experimental trial was over 4×10^6 rounds. The minimax a.r.p.s. for the Attacker is ~ 21.36.

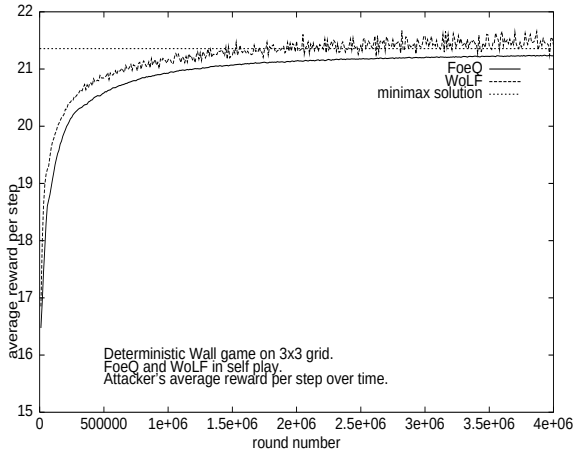


(a) Initial position (b) Attacker passes defender

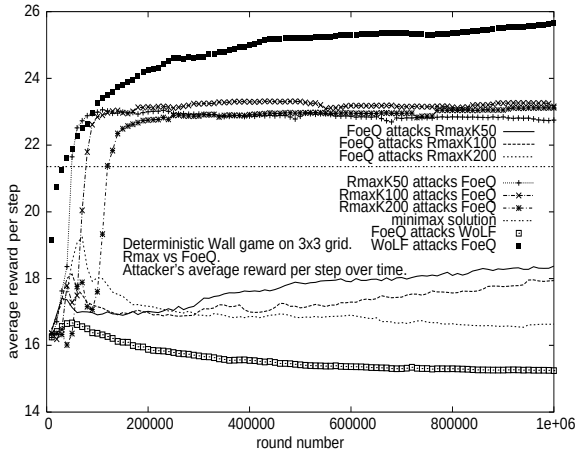
Figure 11: 3x3 wall game



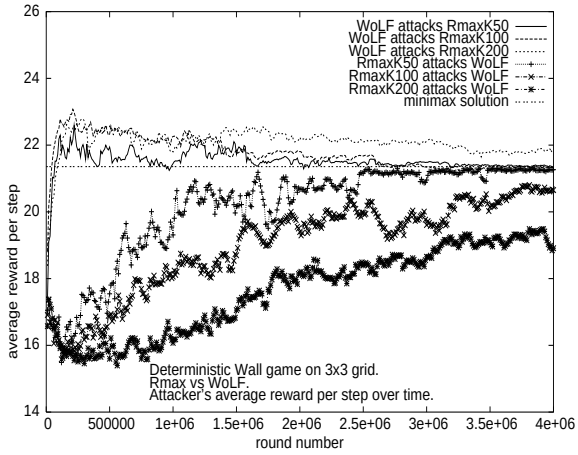
(a) Rmax in self play



(b) FoeQ and WoLF in self play



(c) Rmax vs FoeQ and WoLF vs FoeQ (**only first** 10^6 **rounds**)



(d) Rmax vs WoLF

Figure 12: 3x3 wall game – average reward per step

			Attacker's policy							Defender's policy						
A	*D*	*K*1	u	l	d	r	s	Q-val	KE	u	l	d	r	s	Q-val	KE
Opt	**Opt**		**.514**	**.0**	**.0**	**.486**	**.0**	**1053**	**1200**	**.0**	**.383**	**.0**	**.617**	**.0**	**947**	**1200**
RX	RX	50	.514	.0	.0	.486	.0	—	838	.0	.383	.0	.617	.0	—	838
RX	RX	100	.514	.0	.0	.486	.0	—	833	.0	.383	.0	.617	.0	—	833
RX	RX	200	.515	.0	.0	.485	.0	—	831	.0	.385	.0	.615	.0	—	831
FQ	FQ		.415	.0	.297	.288	.0	966	—	.0	.383	.0	.327	.290	947	—
WF	WF		.279	.006	.279	.432	.004	1095	—	.067	.309	.022	.278	.324	966	—
RX	FQ	50	.213	.0	.181	.606	.0	—	391	.087	.022	.0	.525	.366	265	—
RX	FQ	100	.382	.0	.225	.393	.0	—	295	.054	.066	.068	.597	.215	452	—
RX	FQ	200	.154	.0	.241	.605	.0	—	249	.074	.023	.009	.630	.264	715	—
FQ	RX	50	.040	.094	.0	.727	.139	204	—	.026	.484	.017	.473	.0	—	638
FQ	RX	100	.018	.084	.033	.800	.065	338	—	.048	.683	.075	.194	.0	—	516
FQ	RX	200	.009	.115	.005	.847	.025	517	—	.033	.810	.037	.120	.0	—	391
RX	WF	50	.262	.0	.262	.476	.0	—	583	.06	.147	.013	.334	.446	958	—
RX	WF	100	.388	.0	.170	.442	.0	—	579	.109	.043	.027	.419	.402	997	—
RX	WF	200	.442	.025	.348	.185	.0	—	510	.253	.031	.142	.237	.337	1055	—
WF	RX	50	.238	.0	.185	.577	.0	1061	—	.0	.384	.004	.426	.186	—	487
WF	RX	100	.214	.0	.231	.555	.0	1060	—	.002	.371	.0	.532	.095	—	411
WF	RX	200	.314	.0	.303	.378	.005	1082	—	.026	.401	.010	.524	.039	—	365

Table 4: 3×3 wall game – The first row reports the action probabilities of a NE policy profile in the initial state, the Q-values for action $\langle stand, stand \rangle$ in the initial state and the number of entries in the game. The next rows classify the average learned policies in the initial state, the average learned Q-values for actions $\langle stand, stand \rangle / \langle stand \rangle$ by the FoeQ/WoLF players, respectively, in the initial state and the average number of known entries by Rmax, after 4×10^6 rounds, according to the players' types. RX, FQ and WF are abbreviations for Rmax, FoeQ and WoLF respectively. The first column, titled *A*, provides the Attacker's type. The second column, titled *D*, provides the Defender's type. The third column, titled $K1$, states the value of Rmax's $K1$ parameter. The columns titled *u*, *l*, *d*, *r*, *s* specify the probabilities for actions *up, left, down, right,* and *stand*, respectively, according to the learned policies. The columns titled *Q-val* and *KE* specify the learned Q values by FoeQ or WoLF and the number of known entries by Rmax.

Figure 12 presents the a.r.p.s. obtained by the different agents playing the Attacker's role in self and heterogeneous play.[13] Table 4 classifies some significant variables of the average state of the learning algorithms after 4×10^6 rounds according to the players' types.

Self Play

In self play, all algorithms converge to minimax or almost minimax values (Fig. 12a,b). FoeQ converges to within 0.5 of the minimax value within the first 10^6 rounds and from then on improves very slowly because of its increasing bias to exploit combined with its decreasing learning rates. The FoeQ Defender learns correct Q-values and an optimal policy while the Attacker learns a rough estimation of the Q-values and a suboptimal policy (Table 4).[14] WoLF converges to the minimax value within 1.5×10^6 rounds and then oscillates around this value while the players keep updating

13. In figures where the behavior does not change after a certain point, we show only the initial phases. For example, in Figure 12c).

14. The first row of Table 4 presents the policies as outputted by a value-iteration solver. It should be noted that there are equivalent optimal policies in the initial state: i. For the Attacker, a probability mass of .514 may be divided in any way between the actions *up* and *down*; ii. For the Defender, the actions *right* and *stand* are equivalent since it is next to the right border of the grid, and for the Attacker, *left* and *stand* are equivalent.

their best responses to each other (Fig. 12b). WoLF learns almost optimal policies and Q-values (Table 4). Three main differences make WoLF more robust than FoeQ: first, it maintains a considerably smaller state-action space since it considers only private actions. This difference results in more efficient back propagation of the Q-values. Second, its learning policy is a best response policy rather than an equilibrium policy. For this reason, it collects higher rewards during early phases of learning and in turn the Q-values converge faster. And third, the hill climbing policy updates combined with the variable learning rate serve as an additional exploration mechanism. As long as the WoLF players have not converged to equilibrium, an increased policy learning rate will always be used by one of the players. As in the CISG case, Rmax's learning period depends almost linearly on $K1$. With $K1 = 50$, Rmax converges to the minimax value within 10^6 rounds (Fig. 12a). Unlike the CISG case, Rmax converges to optimal policies before all entries become *known* (Table 4), thus the unknown entries will not be (unnecessarily) further explored.

In self play, the identical exploration and exploitation techniques of both players gives rise to efficient *joint* exploration and hence to fast convergence to policies that are close or equal to the minimax policies. In contrast, when the opponents employ different learning algorithms, joint exploration is impeded.

Heterogeneous Players

Figure 12c depicts the average learning curves for plays of FoeQ against Rmax. The curves start at ~ 16.25, which is the a.r.p.s. for the Attacker when random policies are played. The learning curve for the FoeQ Attacker may be divided into three phases: in the first 50,000 to 100,000 rounds (depending on $K1$), FoeQ's a.r.p.s. increases rapidly, then drops back down to ~ 17 and changes very slowly thereafter. During the first phase, Rmax plays exploratory policies that enable FoeQ to reach its goal states by playing suboptimal policies. As a result, FoeQ propagates Q-values of entries that are not frequently visited by the NE strategies of the game and constructs a wrong estimate of the strategic structure of the game. During the next phase, Rmax learns improved strategies. At this stage, FoeQ is too biased to exploitation and the learning rates for some entries are too small to overcome the distorted estimation in the first phase. In the third phase, new entries rarely become known because of FoeQ's bias to exploit and its slow learning. For lower values of the $K1$ parameter in Rmax, FoeQ yields lower return in the first phase but recovers faster in the third phase because the first phase is shorter for lower values of $K1$. This property in turn results in lower estimation of the Q-values (Table 4) and hence smaller gaps between the true strategic structure of the game and the strategic structure estimated by FoeQ after the first phase. The smaller gaps are easier to overcome in the third phase. The learning curves and learned values and policies for the opposite mode, Rmax Attacks FoeQ, are a bit less distinct but express similar dynamics. The main advantages of Rmax over FoeQ, expressed in the results, are: An exploration technique that is not time dependent instead of increasing greediness, exhaustive computations instead of single backup per step, and the Rmax learning technique that is guaranteed to either explore or attain return that is at least near the minimax, instead of heuristic exploration. Since, from some early stage, FoeQ mainly attempts to exploit w.r.t. its inaccurate estimation of Q-values and strategic structure and since unknown entries are estimated by FoeQ as having low Q-values, the joint policy does not frequently reach unknown entries but rather yields high rewards for Rmax.[15] Figure 12c also shows the learning curves for

15. It is known that Q-learning with low initial values can behave sub-optimally. Thus, the observed behavior of FoeQ may be a consequence of the fact that Q-values are initialized to 0. However, unless prior information about the possible rewards or their magnitude is available, this appears to be the most unbiased choice.

plays of FoeQ against WoLF. The combination of FoeQ's slow learning, fast "aging" and playing an equilibrium policy w.r.t. its pessimistic estimated Q-values with WoLF's fast learning of a good response produces very poor performance for FoeQ in our particular set of experiments.

The average learning curves for plays of WoLF against Rmax, presented in Figure 12d, converge to the minimax value within the examined time interval under the following settings: Rmax Attacks and $K1 = 50$, Rmax Defends and $K1 = 50$ or $K1 = 100$. Prior to convergence, WoLF gains return higher than the minimax. In the cases of convergence to minimax value, Rmax also converges to, at least almost, a NE policy. WoLF in these cases does not converge to NE but rather to some other best response (Table 4). The dynamic of the learning process can roughly be divided into stages, defined by the discovery of new entries by Rmax and the following policy updates. Rmax starts off with a uniformly mixed policy. Before new entries become known to Rmax, WoLF learns a deterministic response policy with a higher return to WoLF than the minimax value. In turn, entries associated with WoLF's deterministic policy are the first to become known to Rmax. Each joint policy that results from Rmax's policy updates has one of the following properties: (i) The new joint policy seldom visits unknown entries. This policy provides Rmax with return close to or greater than the minimax. (ii) The new joint policy frequently visits unknown entries and provides Rmax with higher return than its average so far. (iii) The new joint policy frequently visits unknown entries and provides Rmax with return equal or lower than its average return so far. In cases (i) and (ii), WoLF will switch to "learn fast" mode and, unless Rmax is already playing the minimax policy, will manage to learn a new response policy with return higher than the minimax before the next stage. This joint policy is guaranteed to visit unknown entries but directs exploration to entries more profitable to WoLF. WoLF's hill climbing method, variable learning rate and small action space enable it to adjust fast to Rmax's new policies and maintain an average return higher than the minimax until Rmax converges to the NE policy. Since Rmax starts off with highly exploratory policies, WoLF is able to attain high payoffs at an early learning phase and thus maintain a high threshold for determining switching of learning rates. By playing a best response, WoLF directs joint exploration as to delay convergence of Rmax on one hand and encourage fast growth of its estimated Q-values on the other.

The relation between the value of $K1$ in Rmax to the convergence time is not as clear as in self play because higher values for $K1$ give WoLF more time to adapt and exploit each new policy of Rmax. It should be noted that for deterministic models, $K1$ can be set to 1, leading to very rapid convergence of Rmax. However, our parameter selection attempts to optimize for a wide range of environmental dynamics and assumes this dynamic is not known ahead of time.

The phenomena observed in this game were repeated in the games described in the following sections. Therefore, in the following, only phenomena not observed in the 3×3 Wall game are discussed.

4.5.2 5×2 WALL GAME

The transition rules of this game are identical to the 3×3 Wall game. It is played on a 5×2 grid under deterministic transition probabilities. Figure 13 depicts the initial position of the game and A's reward structure. When A reaches the right column of the grid in row i, it is rewarded by r_i, where $r_0 = 100$ and $r_i = r_{i-1} + 10i$ for $i \in \{1,2,3,4\}$, and the agents are reset in their initial positions. Otherwise A is rewarded by 90. The fixed sum of the game is 200. The minimax a.r.p.s. for the Attacker is ~ 105.18. The game is designed to fool GLIELPs with random exploration played by

A. If *A* explores randomly, it will probably discover the high rewards in the top rows before it will discover the higher rewards in the lower rows. Later, its growing bias to exploit will prevent it from sufficiently exploring the lower rows.

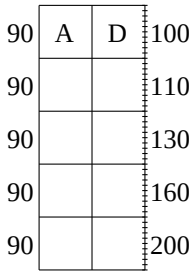
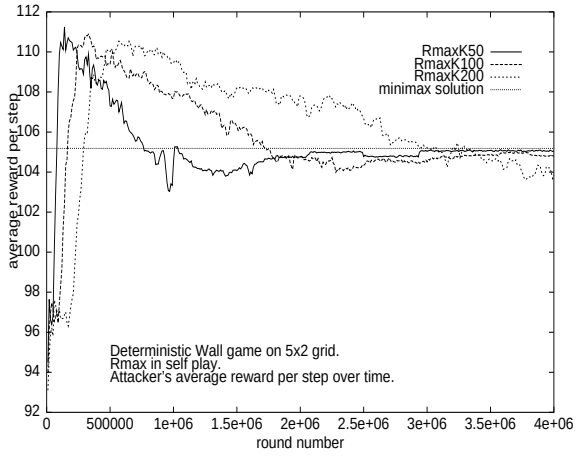
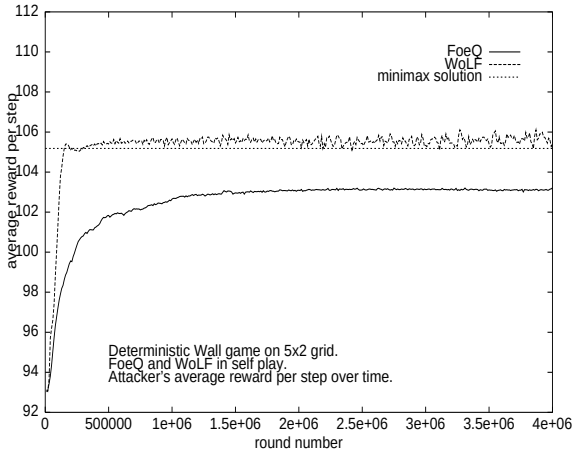


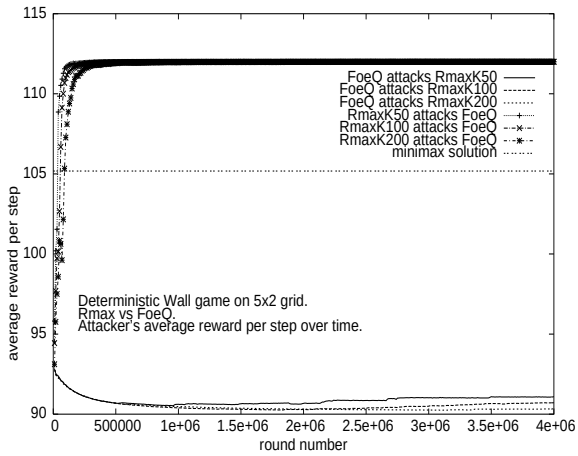
Figure 13: 5×2 wall game – initial position and rewards



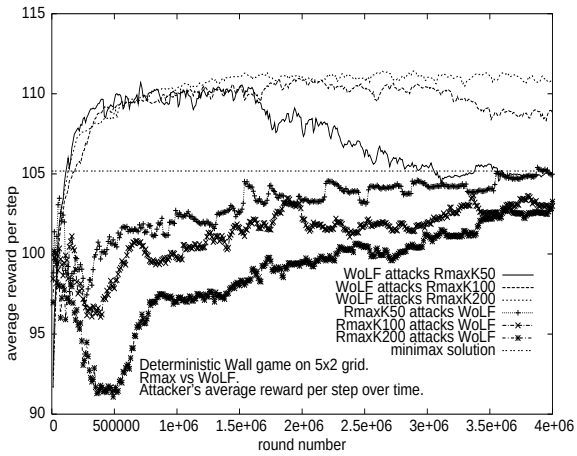
(a) Rmax in self play



(b) FoeQ and WoLF in self play



(c) Rmax vs FoeQ



(d) Rmax vs WoLF

Figure 14: 5×2 wall game – average reward per step

			Attacker							Defender						
A	*D*	*K*1	u	l	d	r	s	Q-val	KE	u	l	d	r	s	Q-val	KE
Opt	**Opt**		**.0**	**.0**	**1.0**	**.0**	**.0**	**5209**	**1125**	**.0**	**.0**	**1.0**	**.0**	**.0**	**4790**	**1125**
RX	RX	50	.0	.0	1.0	.0	.0	—	761	.0	.0	1.0	.0	.0	—	761
RX	RX	100	.0	.0	.971	.029	.0	—	745	.0	.0	1.0	.0	.0	—	745
RX	RX	200	.002	.005	.803	.185	.005	—	723	.0	.051	.903	.046	.0	—	723
FQ	FQ		.648	.0	.0	.306	.045	4783	—	.897	.0	.0	.053	.050	4403	—
WF	WF		.0	.003	.938	.057	.002	5316	—	.001	.0	.999	.0	.0	4742	—
RX	FQ	50	.0	.0	1.0	.0	.0	—	272	.925	.026	.004	.036	.009	507	—
RX	FQ	100	.0	.004	.970	.026	.0	—	229	.949	.006	.0	.023	.022	586	—
RX	FQ	200	.049	.053	.473	.075	.350	—	199	.807	.024	.025	.091	.053	728	—
FQ	RX	50	.213	.081	.158	.293	.255	1726	—	.407	.492	.026	.025	.050	—	88
FQ	RX	100	.252	.193	.014	.319	.222	2861	—	.297	.652	.007	.031	.013	—	58
FQ	RX	200	.312	.165	.033	.318	.172	3926	—	.055	.931	.007	.007	.0	—	43
RX	WF	50	.0	.029	.971	.0	.0	—	456	.0	.0	1.0	.0	.0	4795	—
RX	WF	100	.0	.056	.813	.117	.014	—	438	.0	.0	.992	.004	.004	4832	—
RX	WF	200	.0	.014	.718	.268	.0	—	406	.0	.007	.916	.025	.052	4863	—
WF	RX	50	.0	.0	.996	.004	.0	5223	—	.0	.0	1.0	.0	.0	—	449
WF	RX	100	.025	.049	.856	.021	.050	5400	—	.133	.059	.675	.106	.026	—	361
WF	RX	200	.181	.083	.619	.064	.053	5475	—	.176	.213	.391	.084	.136	—	221

Table 5: 5×2 wall game – NE policies and Average learned policies for state $\langle (2,0),(2,1) \rangle$, average learned Q-values for action $\langle stand, stand \rangle / \langle stand \rangle$ in the initial state by FoeQ/WoLF, respectively, and average number of known entries by Rmax, after 4×10^6 rounds, classified by players' types. See format explanation in Table 4.

Figure 14 presents the a.r.p.s. obtained by the different agents playing the Attacker's role. Table 5 classifies the average learned policies in state $\langle (2,0),(2,1) \rangle$ (both players on middle row), the average learned values for action $\langle stand, stand \rangle / \langle stand \rangle$ in the initial state by the FoeQ/WoLF players and the average number of known entries by the Rmax players, according to the agents playing *A* and *D*. The minimax policy for *A* and *D* when they are both in rows 0, 1 or 2 is to move *down*. When *A* and *D* are both in rows 3 or 4 their minimax policy is mixed.

Self Play

Rmax with $K1 = 50$ converges to the minimax values (Fig. 14a) and policies (Table 5). With values of 100 and 200 of $K1$, Rmax still has a small exploration bias after 4×10^6 rounds (Table 5) and attains almost the minimax value (Fig. 14a). WoLF's convergence to near the minimax value is exceptionally fast in this domain, despite its GLIELP (Fig. 14b). The gradual increase in payoffs for attacking in lower rows both guides exploration and speeds up learning by causing many switches in the learning rate. By employing a high policy learning rate to find a best response, the WoLF Attacker very quickly discovers the high return in the bottom rows and the WoLF Defender follows by defending them. FoeQ on the other hand falls in the designed trap and converges to a suboptimal policy (Fig. 14b). The FoeQ Attacker believes that it can gain higher rewards in the top rows and assigns a high probability to action *up* in state $\langle (2,0),(2,1) \rangle$ instead of *down*. The FoeQ defender, as well, believes that the upper rows are more worth defending and also assigns a high probability to action *up* in this state (Fig. 14b, Table 5).[16]

Heterogeneous Players

Figure 14c shows *A*'s a.r.p.s. over time for plays of FoeQ against Rmax. FoeQ's behavior is

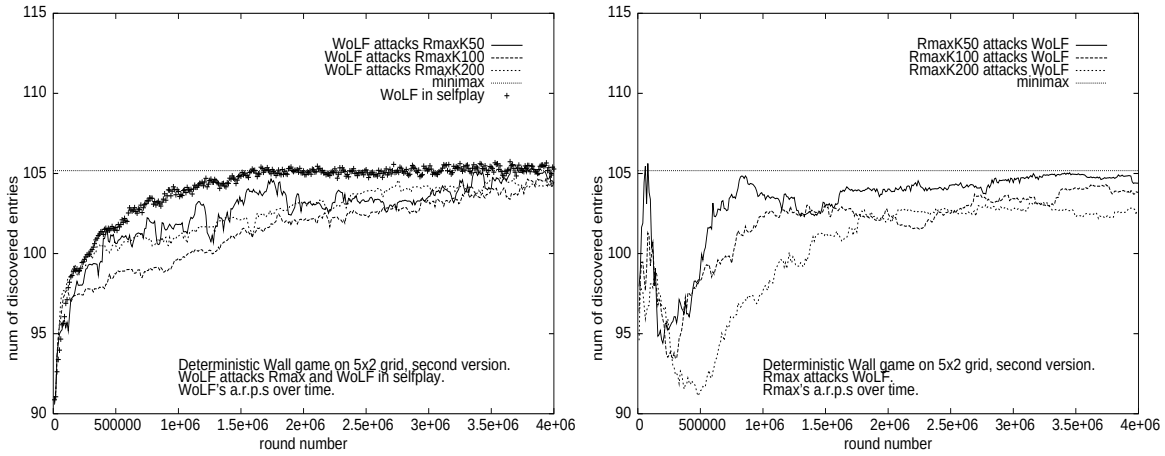
16. Again, this behavior, too, could be attributed to low initial Q-values.

similar to its behavior in self play. When D is played by FoeQ, it attempts to defend the top rows while Rmax attacks in the bottom ones. When A is played by FoeQ, it attempts to Attack only in the top rows (Table 5). Since the entries associated with states of both players being in the bottom rows are unknown to the Rmax Defender, they are modeled as unrewarding by FoeQ, and hence, its policy defends only in the top rows. The low numbers of known entries to Rmax (Table 5) is evidence of FoeQ's inability to sufficiently explore.

In plays of Rmax against WoLF, the players converge to minimax in the examined time interval only for Rmax with $K1 = 50$ and WoLF gains a.r.p.s. higher than the minimax all the way to convergence (Fig. 14d). During the learning period, WoLF's policy is "one step ahead" of Rmax's. In particular, it assigns a greater probability to action *down* in the middle row. This advantage is also expressed by the differences in the probability assigned to action *down* between the learned policies for the different values of $K1$ (Table 5). When WoLF plays the Defender, it adjusts quickly to the behavioral changes of Rmax. When WoLF plays the Attacker, the gradual increase of rewards for attacking in lower rows directs WoLF's exploration while maintaining a high a.r.p.s.

Removing the Intermediate Rewards

To eliminate the incentive to explore and learn quickly given to WoLF by the intermediate rewards, we studied a variant of the 5×2 Wall game: the Attacker's payoffs for attacking in the second and third rows are modified to 100. All the other payoffs and transitions are unchanged. The minimax policies and values are also unchanged since the attacker's optimal policy in both settings attacks only in the two bottom rows.



(a) WoLF attacks Rmax and WoLF in self-play

(b) Rmax attacks WoLF

Figure 15: Modified 5×2 wall game – a.r.p.s. over time

The convergence of WoLF in self play (Fig. 15a) is slower than its convergence on the first reward structure, yet still faster than the other algorithms in self play. In plays of WoLF against Rmax, WoLF has a disadvantage when it plays as the Attacker: The threshold for switching learning rates does not grow in early learning and hence a low learning rate is more frequent while more exploration is required to discover the benefits of attacking in the bottom rows. As an Attacker, WoLF gains a lower return than the minimax value all the way to convergence (Fig. 15a). In the

Defender's role, WoLF's variable learning rate responds to the Rmax Attacker's "initiatives," and WoLF gains higher return than the minimax value all the way to convergence (Fig. 15b).

4.5.3 2×4 TAG GAME

This section describes the results of executing the algorithms on a *stochastic* Tag game. The game is played on a 2×4 grid with a missing corner. One of the players, C, is the tagger and the other, E, is the Escaper. Fig. 16 depicts the initial configuration of the game. A tagging event (tag) occurs when both players have the same positions. In the case of a tag, C receives a reward of 40, E receives a reward of 0 and the players' positions are unchanged. Otherwise C's reward is 15 and E's reward is 25. C's a.r.p.s. under the minimax policy is ~ 18.22. C's optimal strategy is to attempt to trap E in the rightmost cell of the grid while E's optimal strategy is to avoid this situation. To this end, the minimax strategy is deterministic at all states except $\langle (0,2),(0,3) \rangle$.[17] For example, in state $\langle (0,1),(1,1) \rangle$ the Escaper should move *left* and not *right* to avoid the danger of being forced to the corner.

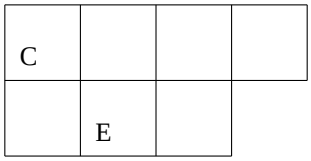


Figure 16: 2x4 tag game – initial position

Figure 17 presents the a.r.p.s. obtained by the different agents playing the tagger's role. Table 6 classifies the average learned policies in state $\langle (0,1),(1,1) \rangle$, the average learned values for action $\langle stand, stand \rangle$ / $\langle stand \rangle$ in state $\langle (0,2),(0,3) \rangle$ by the FoeQ/WoLF players, and the average number of known entries by the Rmax players, after 4×10^6 rounds, according to the agents playing C and E.

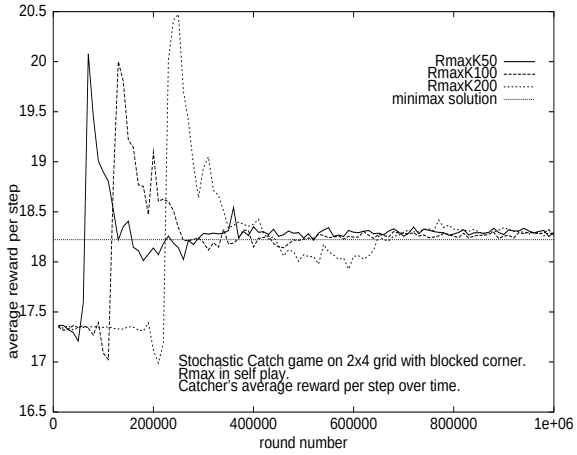
Self Play

In self play, the Rmax Escaper does not learn an optimal policy. Furthermore, in contrast to the deterministic games, E's policy improves with greater values of $K1$, although fewer entries become known (Table 6).[18] This is because more sampling is required to approximate the transition probabilities. However, the learned policies yield an 0.2-optimal return (Fig. 17a), closer to the optimal value than the other algorithms. FoeQ and WoLF converge to near the minimax value within the first 6×10^5 rounds and both learn almost optimal/minimax policies. FoeQ performs much better in this domain than in the previous deterministic domains because the stochastic transitions amplify exploration.

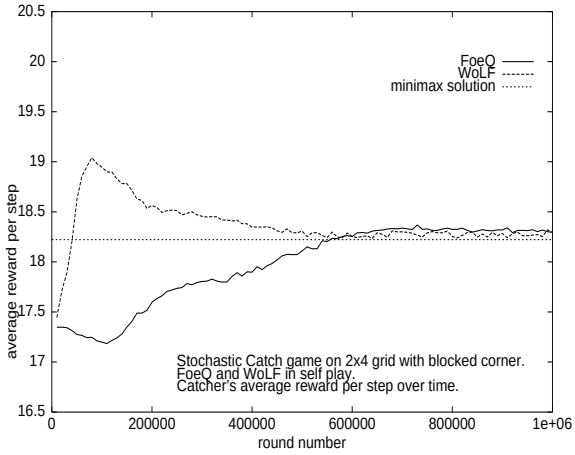
Heterogeneous Players

When Rmax plays against FoeQ, more entries become known and FoeQ's value estimates are better compared to the deterministic games, again due to the amplification of exploration by the

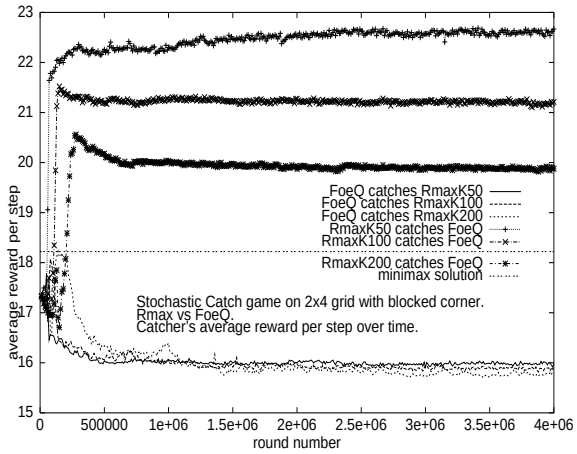
17. Positions are denoted *(row,column)*, with (0,0) being the upper left position. From state $\langle (0,2),(0,3) \rangle$ the players can transit to state $\langle (0,3),(0,2) \rangle$ by the joint action $\langle$*right,left*$\rangle$ without the occurrence of a tag.
18. Recall that as $K1$ increases, more visits are required to mark an entry known. Therefore, fewer entries will be marked within a given time frame.



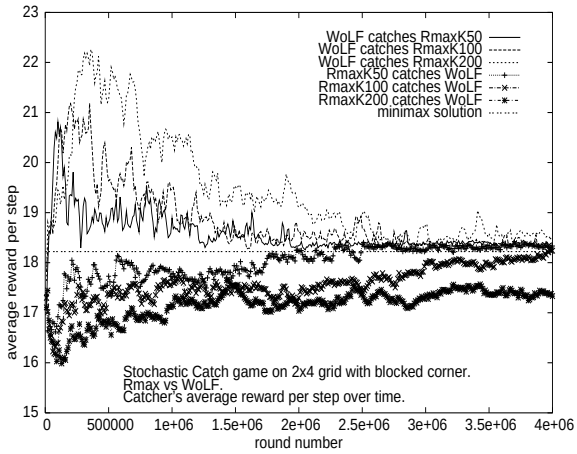
(a) Rmax in self play **(first** 10^6 **rounds)**



(b) FoeQ and WoLF in self play**(first** 10^6 **rounds)**



(c) Rmax vs FoeQ



(d) Rmax vs WoLF

Figure 17: 2×4 tag game – average reward per step

stochastic environmental dynamic. When FoeQ plays Escaper against Rmax, FoeQ learns better policies when Rmax uses larger values of $K1$ (Table 6) and receives a greater average reward (Fig. 17b)—opposite to what was observed in the 3×3 Wall Game. It seems that the longer periods of Rmax playing fixed policies enable FoeQ to better approximate the different Q-values associated with that policy. Despite the stochastic nature of the environment playing "in favor" of FoeQ, Rmax is still superior in heterogeneous play. In plays of Rmax against WoLF, the algorithms converge to the minimax value in five out of the six different configurations, whereas in the deterministic games they converged in two or three out of the six. The convergence dynamic is similar to that observed in the first two games, and WoLF receives an a.r.p.s. greater than the minimax value all the way to convergence.

			Tagger							Escaper						
A	*D*	*K*1	u	l	d	r	s	Q-val	KE	u	l	d	r	s	Q-val	KE
Opt	**Opt**		**.0**	**.0**	**1.0**	**.0**	**.0**	**956**	**1050**	**.0**	**1.0**	**.0**	**.0**	**.0**	**1044**	**1050**
RX	RX	50	.0	.0	1.0	.0	.0	—	844	.0	.450	.0	.550	.0	—	844
RX	RX	100	.0	.0	1.0	.0	.0	—	838	.0	.750	.0	.250	.0	—	838
RX	RX	200	.0	.0	1.0	.0	.0	—	833	.0	.850	.0	.150	.0	—	833
FQ	FQ		.0	.0	.995	.005	.0	889	—	.002	.769	.0	.229	.0	1048	—
WF	WF		.0	.0	1.0	.0	.0	922	—	.005	.655	.0	.340	.0	1089	—
RX	FQ	50	.0	.0	.950	.050	.0	—	742	.138	.278	.164	.225	.195	566	—
RX	FQ	100	.0	.0	1.0	.0	.0	—	649	.067	.150	.064	.618	.101	802	—
RX	FQ	200	.0	.0	1.0	.0	.0	—	572	.038	.320	.061	.541	.040	952	—
FQ	RX	50	.090	.094	.476	.120	.220	448	—	.0	.750	.0	.025	.0	—	789
FQ	RX	100	.112	.029	.471	.049	.339	618	—	.0	.575	.0	.413	.012	—	711
FQ	RX	200	.175	.051	.539	.020	.215	720	—	.0	.461	.001	.438	.100	—	622
RX	WF	50	.0	.0	1.0	.0	.0	—	583	.020	.668	.0	.317	.0	1050	—
RX	WF	100	.0	.0	.983	.0	.017	—	547	.015	.601	.002	.383	.0	1063	—
RX	WF	200	.0	.050	.850	.0	.010	—	488	.020	.392	.001	.531	.056	1098	—
WF	RX	50	.0	.0	.992	.0	.008	973	—	.0	.7	.0	.3	.0	—	608
WF	RX	100	.003	.001	.994	.0	.002	977	—	.0	.65	.0	.35	.0	—	567
WF	RX	200	.032	.019	.928	.008	.013	992	—	.0	0.7	.0	0.3	.0	—	508

Table 6: 2×4 tag game – NE policies and average learned policies for state $\langle (0,1),(1,1) \rangle$, average Q-values for action $\langle stand, stand \rangle$ / $\langle stand \rangle$ in state $\langle (0,2),(0,3) \rangle$ by FoeQ/WoLF and average number of known entries by Rmax, after 4×10^6 rounds, classified by players' types. See format explanation in Table 4.

4.5.4 SUMMARY

The adversarial exploration/exploitation tradeoff in FSSGs is more complicated than that observed in the common interest CISG case. Optimizing behavior during learning introduces a tradeoff between exercising opponents' exploration in order to gain higher return (may-be at the expense of fast convergence to some fixed learning target), to exercising opponents' exploration for joint exploration. When one algorithm takes the time to explore, the other algorithm can exploit and obtain payoff higher than the NE. To this end, learning a best response proves better than learning a NE, when combined in the WoLF algorithm with other properties that ensure fast adaptation to a changing adversary.

Indeed, WoLF appears to be the preferred algorithm in heterogeneous play, with good performance in self-play as well. Nevertheless, WoLF fails to converge to NE in heterogeneous play against an adversary that does converge to a NE, which may be its Achilles heel. WoLF's robustness makes up for the classic weakness of GLIELPs discussed in Section 3.4 (that is, the great sensitivity to the exploration schedule in some domains), but not completely. Thus, while in most cases WoLF is preferable over the other presented algorithms, in some situations this anomaly manifests itself and Rmax outperforms WoLF.

Additional practical issues may affect the choice of algorithm for a specific task. WoLF is computationally more efficient, mainly because it does not involve equilibrium computations. Rmax is much simpler for pre-tuning, with a single intuitive parameter, but requires solving the underlying stochastic game.

5. MGS

MGS is a Markov Game Simulation system designed to evaluate online performance of MARL algorithms. Three main software components take part in a simulation:

Players – user-defined implementations of MARL algorithms.

Referee – a user-defined program that represents a multi-agent environment.

Simulator – mediates between the Players and Referee.

MGS provides Java interfaces and an abstract Referee class that implements the backbone of typical grid-world environments and makes the programming of grid worlds simple and easy. It should be noted that the description of software components and methods in the rest of this section is for illustrative purposes and is partial and incomplete.

Modeling real world environments (or simplifications of such environments) as Stochastic Games is a tedious task for humans. To simplify the modeling task, MGS supports simple creation of grid-world environments referred to as Grid Games (GG). In a GG, agents can move about between squares of a grid, move/carry objects etc. GGs induce MGs in which the set of states S are the possible assignments to the state variables, which are typically the position of the agents and various objects. Actions change the positions of the agents and the state of the objects.

5.1 The Referee

This program represents a GG. The state variables of the GG are memory variables of the Referee program and reachable internal states of the Referee correspond to possible assignments to the state variables. The Referee may manage additional memory variables, that is, variables that capture the previous assignment to the state variables in order to implement the payoff function. The Referee implements methods that simulate the environment such as:

- `getStateIndex()` - enumerates the state space. Returns a unique integer that corresponds to the current state of the Referee.
- `giveActions(int[] actions)` - receives the action choices of the players and updates the state variables to characterize the new state of the environment.
- `getPlayerReward(int p)` - returns the payoff for player `p`.

5.2 The Players

They implement the methods:

- `play(int s)` - returns the action choice in state `s`.
- `update(int s, int[] acts, double r)` - updates the algorithm's model / values according to the new state `s`, other Players' actions `acts` and payoff `r`.

5.3 The Simulator

This module is the active process during simulation. Schematically, the Simulator loops over the following steps:

1. get the Players' actions in the current state.
2. pass the joint action to the Referee.
3. compute the index of the new state of the environment.
4. pass the new state index and payoffs to the players.

Typically, GGs involve actions that move the agents up, down, left and right on a two-dimensional grid. To unburden the user from modeling these aspects of the environment, they are already built into the system. The abstract Referee class implements various methods for manipulating the positions of the agents on a grid represented by a two dimensional integer array. The Simulator also computes the new positions of the agents according to the five default actions `up`, `left`, `down`, `right` and `stand`, and according to user input transition probabilities. In Step 2 above, the Simulator also passes to the Referee the results of this computation in the form of suggested new positions for the players.

MGS is a very flexible tool. Despite the implemented GG features, it can in fact be used to model *any* discrete state-action space MG (although doing so may require more complicated programming than the simple implementation of GGs). MGS offers various features that make it a convenient experimental tool. Input can be specified either by a GUI or by a script. Scripts may specify multiple independent simulations and may also include parameters for the Players' algorithms. MGS logs statistics of payoffs and selected actions and also supports logging by the Players. For further information on MGS, see `http://www.cs.bgu.ac.il/~mal`.

6. Conclusions

This paper presents a large empirical study of representative MARL algorithms conducted using the MGS tool. Such comprehensive studies in this area are rare. The only other related study we are aware of appeared in Powers and Shoham (2005) and involved the much simpler class of repeated games with known game matrices. While most authors run some empirical studies, these often focus on their algorithms and do provide a comprehensive comparison of strengths and weaknesses.

We believe that our results and analysis can serve to guide researchers in developing more powerful algorithms and formal analytic tools, and practitioners in selecting and tuning algorithms for specific tasks. Some of our results are closely related to phenomena observed in single-agent reinforcement-learning algorithms, especially in common-interest environments, which can be viewed as describing a distributed version of single-agent RL. In this domain, the issue of exploration vs. exploitation appears to play a major role in the success of different algorithms. Here, we found Rmax's exhaustive approach to be very useful, being much less susceptible to being stuck in local minima compared to GLIELP exploration. Of course, one does expect this almost-exhaustive exploration approach to be costly in large domains. However, in our examples, Rmax was able to perform well with small sample sizes, partly due to the locality of actions—that is, the fact that most actions have a small number of possible outcomes and these effects do not change the agent's state drastically. In general, many real-world domains tend to have this property. We believe that

online identification of locality properties may be used to construct more practical variants of Rmax as well as other methods. Rmax also provides another capability we found important in common interest games: coordinated exploration. It also seems to scale well with the number of agents. Perhaps most important is the fact that it is very simple to understand its behavior, and consequently, we believe, to modify it given background knowledge. However, Rmax is completely inadequate if cooperation is to be obtained given a system with heterogeneous agents.

Fixed sum games provided an interesting setting, where we could test algorithms against each other. We found that learning is more efficient when the greedy component of the learning policy is a best response rather than a minimax strategy. The WoLF algorithm achieves fast adaptation to a changing opponent by maintaining values for only the private action space and by regulating behavior according to the dynamics of the learning process, and it seems to be the best choice in such competitive environments.

Overall, it seems that there is much potential for improved performance by multi-agent learning algorithms. We hope this study will motivate the design of algorithms that improve upon the current state of the art, and we believe that the MGS test bed can be a useful tool for testing such new techniques.

Acknowledgments

We are grateful to the reviewers for their useful suggestions and comments, and to the associate editor, Michael Littman, for his many useful and detailed comments. Partial support was provided by the Paul Ivanier Center for Robotics Research and Production Management, by the Lynn and William Frankel Center for Computer Science, and by the Israel Science Foundation.

References

A. Bab and R. I. Brafman. An experimental study of different approaches to reinforcement learning in common interest stochastic games. In *ECML*, pages 75–86, 2004.

M. H. Bowling and M. M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

R. I. Brafman and M. Tennenholtz. R-max – a general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, 3:213–231, 2002.

R. I. Brafman and M. Tennenholtz. Learning to coordinate efficiently: A model based approach. *JAIR*, 19:11–23, 2003.

R I. Brafman and M. Tennenholtz. Efficient learning equilibrium. *Artif. Intell.*, 159:27–47, 2004.

G. W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, 1951.

G. Chalkiadakis and C. Boutilier. Coordination in multiagent reinforcement learning: A Bayesian approach. In *AAMAS'03*, 2003.

C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multi-agent systems. In *Proc. Workshop on Multi-Agent Learning*, 1997.

R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *UAI'99*, 1999.

R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *AAAI-98*, 1998.

E. Even-Dar and Y. Mansour. Learning rates for *Q*-learning. *Journal of Machine Learning Research*, 5:1–25, 2003.

J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1997.

J. Hu and M.P. Wellman. Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 1095–1100, 1998.

L. P. Kaelbling. *Learning in Embedded Systems*. The MIT Press: Cambridge, MA, 1993.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

M. L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proc. 18th International Conf. on Machine Learning*, 2001.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. 11th International Conference on Machine Learning*, 1994.

A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

R. Powers and Y. Shoham. Learning against opponents with bounded memory. In *Proc. 19th International Joint Conf. on Artificial Intelligence*, 2005.

M. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.

A. Schaerf, Y. Shoham, and M. Tennenholtz. Adaptive load balancing: A study in multi-agent learning. *Journal of Artificial Intelligence Research*, 2:475–500, 1995.

Y. Shoham, R. Powers, and T. Grenager. If Multi-Agent Learning is the Answer, What is the Question? *Artificial Intelligence*, 171(7):365–377, 2007.

S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158, 1996.

M. Sridharan and G. Tesauro. Multi-agent Q-learning and regression trees for automated pricing decisions. In *Proc. 17th International Conf. on Machine Learning*, pages 927–934. Morgan Kaufmann, San Francisco, CA, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

W. Uther and M. Veloso. Adversarial reinforcement learning. Technical report, Carnegie Mellon University, 2003.

J. M. Vidal and E. H. Durfee. Predicting the expected behavior of agents that learn about agents: the clri framework. *Autonomous Agents and Multi-Agent Systems*, 6(1):77–107, 2003.

R. V. Vohraa and M. P. Wellman. Foundations of multi-agent learning: Introduction to the special issue. *Artificial Intelligence*, 7:363–364, 2007.

X. Wang and T. Sandholm. Reinforcement learning to play an optimal nash equilibrium in team markov games. In *NIPS'02*, 2002.

Z. Zheng, M. Shu-gen, C. Bing-gang, Z. Li-ping, and L. Bin. Multiagent reinforcement learning for a planetary exploration multirobot system. In *PRIMA*, pages 339–350, 2006.

Journal of Machine Learning Research 9 (2008) 2677-2694 Submitted 10/07; Revised 4/08; Published 12/08

An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons

Salvador García SALVAGL@DECSAI.UGR.ES
Francisco Herrera HERRERA@DECSAI.UGR.ES
Department of Computer Science and Artificial Intelligence
University of Granada
Granada, 18071, Spain

Editor: John Shawe-Taylor

Abstract

In a recently published paper in JMLR, Demšar (2006) recommends a set of non-parametric statistical tests and procedures which can be safely used for comparing the performance of classifiers over multiple data sets. After studying the paper, we realize that the paper correctly introduces the basic procedures and some of the most advanced ones when comparing a control method. However, it does not deal with some advanced topics in depth. Regarding these topics, we focus on more powerful proposals of statistical procedures for comparing $n \times n$ classifiers. Moreover, we illustrate an easy way of obtaining adjusted and comparable p-values in multiple comparison procedures.

Keywords: statistical methods, non-parametric test, multiple comparisons tests, adjusted p-values, logically related hypotheses

1. Introduction

In the Machine Learning (ML) scientific community there is a need for rigorous and correct statistical analysis of published results, due to the fact that the development or modifications of algorithms is a relatively easy task. The main inconvenient related to this necessity is to understand and study the statistics and to know the exact techniques which can or cannot be applied depending on the situation, that is, type of results obtained. In a recently published paper in JMLR by Demšar (2006), a group of useful guidelines are given in order to perform a correct analysis when we compare a set of classifiers over multiple data sets. Demšar recommends a set of non-parametric statistical techniques (Zar, 1999; Sheskin, 2003) for comparing classifiers under these circumstances, given that the sample of results obtained by them does not fulfill the required conditions and it is not large enough for making a parametric statistical analysis. He analyzed the behavior of the proposed statistics on classification tasks and he checked that they are more convenient than parametric techniques.

Recent studies apply the guidelines given by Demšar in the analysis of performance of classifiers (Esmeir and Markovitch, 2007; Marrocco et al., 2008). In them, a new proposal or methodology is offered and it is compared with other methods by means of pairwise comparisons. Another type of studies assume an empirical comparison or review of already proposed methods. In these cases, no proposal is offered and a statistical comparison could be very useful in determining the differences among the methods. In the specialized literature, many papers provide reviews on a specific topic and they also use statistical methodology to perform comparisons. For example, in a review of

ensembles of decision trees, non-parametric tests are also applied in the analysis of performance (Banfield et al., 2007). However, only the rankings computed by Friedman's method (Friedman, 1937) are stipulated and authors establish comparisons based on them, without taking into account significance levels. Demšar focused his work in the analysis of new proposals, and he introduced the Nemenyi test for making all pairwise comparisons (Nemenyi, 1963). Nevertheless, the Nemenyi test is very conservative and it may not find any difference in most of the experimentations. In recent papers, the authors have used the Nemenyi test in multiple comparisons. Due to the fact that this test posses low power, authors have to employ many data sets (Yang et al., 2007b) or most of the differences found are not significant (Yang et al., 2007a; Núñez et al., 2007). Although the employment of many data sets could seem beneficial in order to improve the generalization of results, in some specific domains, that is, imbalanced classification (Owen, 2007) or multi-instance classification (Murray et al., 2005), data sets are difficult to find.

Procedures with more power than Nemenyi's one can be found in specialized literature. We have based on the necessity to apply more powerful procedures in empirical studies in which no new method is proposed and the benefit consists of obtaining more statistical differences among the classifiers compared. Thus, in this paper we describe these procedures and we analyze their behavior by means of the analysis of multiple repetitions of experiments with randomly selected data sets.

On the other hand, we can see other works in which the *p*-value associated to a comparison between two classifiers is reported (García-Pedrajas and Fyfe, 2007). Classical non-parametric tests, such as Wilcoxon and Friedman (Sheskin, 2003), may be incorporated in most of the statistical packages (SPSS, SAS, R, etc.) and the computation of the final *p*-value is usually implemented. However, advanced procedures such as Holm (1979), Hochberg (1988), Hommel (1988) and the ones described in this paper are usually not incorporated in statistical packages. The computation of the correct *p*-value, or Adjusted *P*-Value (APV) (Westfall and Young, 2004), in a comparison using any of these procedures is not very difficult and, in this paper, we show how to include it with an illustrative example.

The paper is set up as follows. Section 2 presents more powerful procedures for comparing all the classifiers among them in a $n \times n$ comparison of multiple classifiers and a case study. In Section 3 we describe the procedures for obtaining the APV by considering the post-hoc procedures explained by Demšar and the ones explained in this paper. In Section 4, we perform an experimental study of the behavior of the statistical procedures and we discuss the results obtained. Finally, Section 5 concludes the paper.

2. Comparison of Multiple Classifiers: Performing All Pairwise Comparisons

In the paper Demšar (2006), referring to carrying out comparisons of more than two classifiers, a set of useful guidelines were given for detecting significant differences among the results obtained and post-hoc procedures for identifying these differences. Friedman's test is an omnibus test which can be used to carry out these types of comparison. It allows to detect differences considering the global set of classifiers. Once Friedman's test rejects the null hypothesis, we can proceed with a post-hoc test in order to find the concrete pairwise comparisons which produce differences. Demšar described the use of the Nemenyi test used when all classifiers are compared with each other. Then, he focused on procedures that control the family-wise error when comparing with a control classifier, arguing that the objective of a study is to test whether a newly proposed method is better than the existing

ones. For this reason, he described and studied in depth more powerful and sophisticated procedures derived from Bonferroni-Dunn such as Holm's, Hochberg's and Hommel's methods.

Nevertheless, we think that performing all pairwise comparisons in an experimental analysis may be useful and interesting in different cases when proposing a new method. For example, it would be interesting to conduct a statistical analysis over multiple classifiers in review works in which no method is proposed. In this case, the repetition of comparisons choosing different control classifiers may lose the control of the family-wise error.

Our intention in this section is to give a detailed description of more powerful and advanced procedures derived from the Nemenyi test and to show a case study that uses these procedures.

2.1 Advanced Procedures for Performing All Pairwise Comparisons

A set of pairwise comparisons can be associated with a set or family of hypotheses. Any of the post-hoc tests which can be applied to non-parametric tests (that is, those derived from the Bonferroni correction or similar procedures) work over a family of hypotheses. As Demšar explained, the test statistics for comparing the i-th and j-th classifier is

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6N}}},$$

where R_i is the average rank computed through the Friedman test for the i-th classifier, k is the number of classifiers to be compared and N is the number of data sets used in the comparison.

The z value is used to find the corresponding probability (p-value) from the table of normal distribution, which is then compared with an appropriate level of significance α (Table A1 in Sheskin, 2003). Two basic procedures are:

- Nemenyi (1963) procedure: it adjusts the value of α in a single step by dividing the value of α by the number of comparisons performed, $m = k(k-1)/2$. This procedure is the simplest but it also has little power.

- Holm (1979) procedure: it was also described in Demšar (2006) but it was used for comparisons of multiple classifiers involving a control method. It adjusts the value of α in a step down method. Let $p_1, ..., p_m$ be the ordered p-values (smallest to largest) and $H_1, ..., H_m$ be the corresponding hypotheses. Holm's procedure rejects H_1 to $H_{(i-1)}$ if i is the smallest integer such that $p_i > \alpha/(m-i+1)$. Other alternatives were developed by Hochberg (1988), Hommel (1988) and Rom (1990). They are easy to perform, but they often have a similar power to Holm's procedure (they have more power than Holm's procedure, but the difference between them is not very notable) when considering all pairwise comparisons.

The hypotheses being tested belonging to a family of all pairwise comparisons are logically interrelated so that not all combinations of true and false hypotheses are possible. As a simple example of such a situation suppose that we want to test the three hypotheses of pairwise equality associated with the pairwise comparisons of three classifiers $C_i, i = 1, 2, 3$. It is easily seen from the relations among the hypotheses that if any one of them is false, at least one other must be false. For example, if C_1 is better/worse than C_2, then it is not possible that C_1 has the same performance as C_3 and C_2 has the same performance as C_3. C_3 must be better/worse than C_1 or C_2 or the two classifiers at the same time. Thus, there cannot be one false and two true hypotheses among these three.

Based on this argument, Shaffer proposed two procedures which make use of the logical relation among the family of hypotheses for adjusting the value of α (Shaffer, 1986).

- Shaffer's static procedure: following Holm's step down method, at stage j, instead of rejecting H_i if $p_i \leq \alpha/(m-i+1)$, reject H_i if $p_i \leq \alpha/t_i$, where t_i is the maximum number of hypotheses which can be true given that any $(i-1)$ hypotheses are false. It is a static procedure, that is, $t_1, ..., t_m$ are fully determined for the given hypotheses $H_1, ..., H_m$, independent of the observed p-values. The possible numbers of true hypotheses, and thus the values of t_i can be obtained from the recursive formula

$$S(k) = \bigcup_{j=1}^{k} \{ \binom{j}{2} + x : x \in S(k-j) \},$$

  where $S(k)$ is the set of possible numbers of true hypotheses with k classifiers being compared, $k \geq 2$, and $S(0) = S(1) = \{0\}$.

- Shaffer's dynamic procedure: it increases the power of the first by substituting α/t_i at stage i by the value α/t_i^*, where t_i^* is the maximum number of hypotheses that could be true, given that the previous hypotheses are false. It is a dynamic procedure since t_i^* depends not only on the logical structure of the hypotheses, but also on the hypotheses already rejected at step i. Obviously, this procedure has more power than the first one. In this paper, we have not used this second procedure, given that it is included in an advanced procedure which we will describe in the following.

In Bergmann and Hommel (1988) was proposed a procedure based on the idea of finding all elementary hypotheses which cannot be rejected. In order to formulate Bergmann-Hommel's procedure, we need the following definition.

Definition 1 *An index set of hypotheses $I \subseteq \{1, ..., m\}$ is called* exhaustive *if exactly all H_j, $j \in I$, could be true.*

In order to exemplify the previous definition, we will consider the following case: We have three classifiers, and we will compare them in a $n \times n$ comparison. We will obtain three hypotheses:

- H_1 = C_1 es equal in behavior than C_2.
- H_2 = C_1 es equal in behavior than C_3.
- H_3 = C_2 es equal in behavior than C_3.

and eight possible sets S_i:

- S_1: All H_j are true.
- S_2: H_1 and H_2 are true and H_3 is false.
- S_3: H_1 and H_3 are true and H_2 is false.

- S_4: H_2 and H_3 are true and H_1 is false.
- S_5: H_1 is true and H_2 and H_3 are false.
- S_6: H_2 is true and H_1 and H_3 are false.
- S_7: H_3 is true and H_1 and H_2 are false.
- S_8: All H_j are false.

Sets S_1, S_5, S_6, S_7 and S_8 can be possible, because their hypotheses can be true at the same time, so they are exhaustive sets. Set S_2, basing on logically related hypotheses principles, is not possible because the performance of C_1 cannot be equal to C_2 and C_3, whereas C_2 has different performance than C_3. The same consideration can be done to S_3 and S_4, which are not exhaustive sets.

Under this definition, it works as follows.

- Bergmann and Hommel (1988) procedure: Reject all H_j with $j \notin A$, where the *acceptance set*

$$A = \bigcup \{I : I \textit{ exhaustive}, \ min\{P_i : i \in I\} > \alpha/|I|\}$$

  is the index set of null hypotheses which are retained.

  For this procedure, one has to check for each subset I of $\{1,...,m\}$ if I is exhaustive, which leads to intensive computation. Due to this fact, we will obtain a set, named E, which will contain all the possible exhaustive sets of hypotheses for a certain comparison. A rapid algorithm which was described in Hommel and Bernhard (1994) allows a substantial reduction in computing time. Once the E set is obtained, the hypotheses that do not belong to the A set are rejected.

  Figure 1 shows a valid algorithm for obtaining all the exhaustive sets of hypotheses, using as input a list of classifiers C. E is a set of families of hypotheses; likewise, a family of hypotheses is a set of hypotheses. The most important step in the algorithm is the number 6. It performs a division of the classifiers into two subsets, in which the last classifier k always is inserted in the second subset and the first subset cannot be empty. In this way, we ensure that a subset yielded in a division is never empty and no repetitions are produced. For example, suppose a set C with three classifiers $C = \{1,2,3\}$. All possible divisions without taking into account the previous assumptions are: $D_1 = \{C_1 = \{\}, C_2 = \{1,2,3\}\}, D_2 = \{C_1 = \{1\}, C_2 = \{2,3\}\}, D_3 = \{C_1 = \{2\}, C_2 = \{1,3\}\}, D_4 = \{C_1 = \{1,2\}, C_2 = \{3\}\}, D_5 = \{C_1 = \{3\}, C_2 = \{1,2\}\}, D_6 = \{C_1 = \{1,3\}, C_2 = \{2\}\}, D_7 = \{C_1 = \{2,3\}, C_2 = \{1\}\}, D_8 = \{C_1 = \{1,2,3\}, C_2 = \{\}\}$. Divisions D_1 and D_8, D_2 and D_7, D_3 and D_6, D_4 and D_5 are equivalent, respectively. Furthermore, divisions D_1 and D_8 are not interesting. Using the assumptions in step 6 of the algorithm, the possible divisions are: $D_1 = \{C_1 = \{1\}, C_2 = \{2,3\}\}, D_2 = \{C_1 = \{2\}, C_2 = \{1,3\}\}, D_3 = \{C_1 = \{1,2\}, C_2 = \{3\}\}$. In this case, all the divisions are interesting and no repetitions are yielded. The computational complexity of the algorithm for obtaining exhaustive sets is $O(2^{n^2})$. However, the computation requirements may be reduced by means of using storage capabilities. Relative exhaustive sets for $k-i$, $1 \leq i \leq (k-2)$ classifiers can be stored in memory and there is no necessity of invoking the *obtainingExhaustive* function recursively. The computational complexity using storage capabilities is $O(2^n)$, so the algorithm still requires intensive computation.

An example illustrating the algorithm for obtaining all exhaustive sets is drawn in Figure 2. In it, four classifiers, enumerated from 1 to 4 in the C set, are used. The comparisons or hypotheses are denoted by pairs of numbers without a separation character between them. This illustration does not show the case in which the set $|C_i| < 2$, for simplifying the representation. When $|C_i| < 2$, no comparisons can be performed, so the *obtainExhaustive* function returns an empty set E.

An edge connecting two boxes represents an invocation of this function. In each box, the list of classifiers given as input and the first initialization of the E set are displayed. The main edges, whose starting point is the initial box, are labeled by the order of invocation. Below the graph, the resulting E subset in each main edge is denoted. The final E will be composed by the union of these E subsets. At the end of the process, 14 distinct exhaustive sets are found: $E = \{(12,13,14,23,24,34),(23,24,34),(13,14,34),(12,14,24),(12,13,23),$ $(12),(13),(14),(23),(24),(34),(12,34),(13,24),(23,14)\}$.

Table 1 gives the number of hypotheses (m), the number (2^n) of index sets I and the number of exhaustive index sets (n_e) for k classifiers being compared.

Function obtainExhaustive($C = \{c_1, c_2, ..., c_k\}$: list of classifiers)

1. Let $E = \emptyset$
2. $E = E \cup \{$set of all possible and distinct pairwise comparisons using $C\}$
3. If $E == \emptyset$
   4. Return E
5. End if
6. For all possible divisions of C into two subsets C_1 and C_2, $c_k \in C_2$ and $C_1 \neq \emptyset$
   7. $E_1 = obtainExhaustive(C_1)$
   8. $E_2 = obtainExhaustive(C_2)$
   9. $E = E \cup E_1$
   10. $E = E \cup E_2$
   11. For each family of hypotheses e_1 of E_1
      12. For each family of hypotheses e_2 of E_2
         13. $E = E \cup (e_1 \cup e_2)$
      14. End for
   15. End for
16. End for
17. Return E

Figure 1: Algorithm for obtaining all exhaustive sets

The following subsections present a case study of a $n \times n$ comparison of some well-known classifiers over thirty data sets. In it, the four procedures explained above are employed.

2.2 Performing All Pairwise Comparisons: A Case Study

In the following, we show an example involving the four procedures described with a comparison of five classifiers: C4.5 (Quinlan, 1993); One Nearest Neighbor (1-NN) with Euclidean distance,

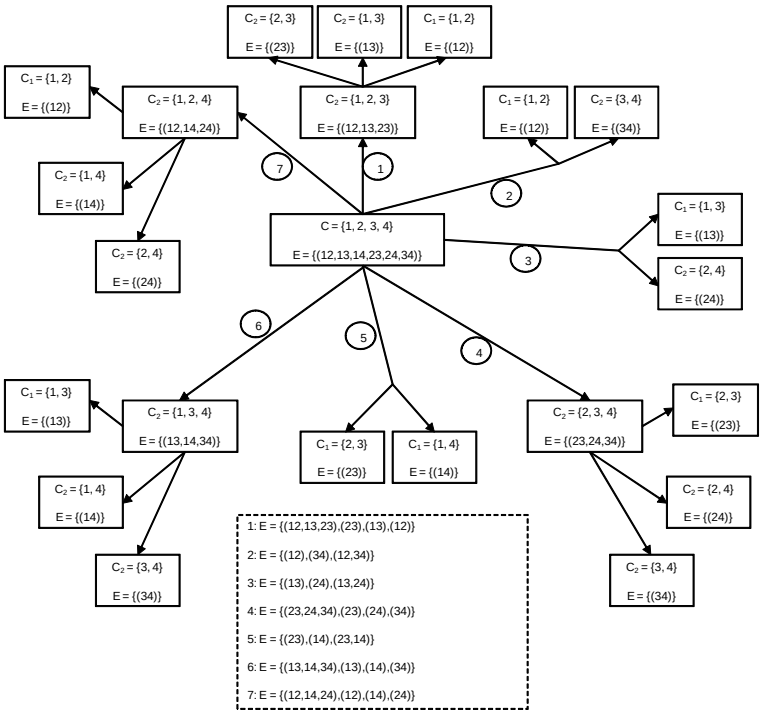


Figure 2: Example of the obtaining of exhaustive sets of hypotheses considering 4 classifiers

k	$m = \binom{k}{2}$	2^m	n_e
4	6	64	14
5	10	1024	51
6	15	32768	202
7	21	2097152	876
8	28	$2.7 \cdot 10^8$	4139
9	36	$6.7 \cdot 10^{10}$	21146

Table 1: All pairwise comparisons of k classifiers

NaiveBayes, Kernel (McLachlan, 2004)[1] and, finally, CN2 (Clark and Niblett, 1989).[2] The parameters used are specified in Section 4. We have used 10-fold cross validation and standard parameters for each algorithm. The results correspond to average accuracy or $1 - class\_error$ in test data. We have used 30 data sets.[3] Table 2 shows the overall process of computation of average rankings.

Friedman (1937) and Iman and Davenport (1980) tests check whether the measured average ranks are significantly different from the mean rank $R_j = 3$. They respectively use the χ^2 and the F statistical distributions to determine if a distribution of observed frequencies differs from the theoretical expected frequencies. Their statistics use nominal (categorical) or ordinal level data, instead of using means and variances. Demšar (2006) detailed the computation of the critical values in each distribution. In this case, the critical values are 9.488 and 2.45, respectively at $\alpha = 0.05$, and the Friedman's and Iman-Davenport's statistics are:

$$\chi^2_F = 39.647, F_F = 14.309.$$

Due to the fact that the critical values are lower than the respective statistics, we can proceed with the post-hoc tests in order to detect significant pairwise differences among all the classifiers. For this, we have to compute and order the corresponding statistics and p-values. The standard error in the pairwise comparison between two classifiers is $SE = \sqrt{\frac{k(k+1)}{6N}} = \sqrt{\frac{5 \cdot 6}{6 \cdot 30}} = 0.408$. Table 3 presents the family of hypotheses ordered by their p-value and the adjustment of α by Nemenyi's, Holm's and Shaffer's static procedures.

- Nemenyi's test rejects the hypotheses [1–4] since the corresponding p-values are smaller than the adjusted α's.
- Holm's procedure rejects the hypotheses [1–5].
- Shaffer's static procedure rejects the hypotheses [1–6].
- Bergmann-Hommel's dynamic procedure first obtains the exhaustive index set of hypotheses. It obtains 51 index sets. We can see them in Table 4. From the index sets, it computes the A set.[4] It rejects all hypotheses H_j with $j \notin A$, so it rejects the hypotheses [1–8].

  Bergmann-Hommel's dynamic procedure allows to clearly distinguishing among three groups of classifiers, attending to their performance:

- Best classifiers: C4.5 and NaiveBayes.
- Middle classifiers: 1-NN and CN2.
- Worst classifier: Kernel.

1. Kernel method is a bayesian classifier which employs a non-parametric estimation of density functions through a gaussian kernel function. The adjustment of the covariance matrix is performed by the ad-hoc method.
2. NaiveBayes and CN2 are classifiers for discrete domains, so we have discretized the data prior to learning with them. The discretizer algorithm is Fayyad and Irani (1993).
3. Data sets marked with '*' have been subsampled being adapted to slow algorithms, such as CN2.
4. We have considered that each classifier follows the order: 1 - C4.5, 2 - 1-NN, 3 - NaiveBayes, 4 - Kernel, 5 - CN2. For example, the hypothesis 13 represents the comparison between C4.5 and NaiveBayes.

	C4.5	1-NN	NaiveBayes	Kernel	CN2
Abalone*	0.219 (3)	0.202 (4)	0.249 (2)	0.165 (5)	0.261 (1)
Adult*	0.803 (2)	0.750 (4)	0.813 (1)	0.692 (5)	0.798 (3)
Australian	0.859 (1)	0.814 (4)	0.845 (2)	0.542 (5)	0.816 (3)
Autos	0.809 (1)	0.774 (3)	0.673 (4)	0.275 (5)	0.785 (2)
Balance	0.768 (3)	0.790 (2)	0.727 (4)	0.872 (1)	0.706 (5)
Breast	0.759 (1)	0.654 (5)	0.734 (2)	0.703 (4)	0.714 (3)
Bupa	0.693 (1)	0.611 (3)	0.572 (4.5)	0.689 (2)	0.572 (4.5)
Car	0.915 (1)	0.857 (3)	0.860 (2)	0.700 (5)	0.777 (4)
Cleveland	0.544 (2)	0.531 (4)	0.558 (1)	0.439 (5)	0.541 (3)
Crx	0.855 (2)	0.796 (4)	0.857 (1)	0.607 (5)	0.809 (3)
Dermatology	0.945 (3)	0.954 (2)	0.978 (1)	0.541 (5)	0.858 (4)
German	0.725 (2)	0.705 (4)	0.739 (1)	0.625 (5)	0.717 (3)
Glass	0.674 (4)	0.736 (1)	0.721 (2)	0.356 (5)	0.704 (3)
Hayes-Roth	0.801 (1)	0.357 (4)	0.520 (2.5)	0.309 (5)	0.520 (2.5)
Heart	0.785 (2)	0.770 (3)	0.841 (1)	0.659 (5)	0.759 (4)
Ion	0.906 (2)	0.359 (5)	0.895 (3)	0.641 (4)	0.918 (1)
Led7Digit	0.710 (2)	0.402 (4)	0.728 (1)	0.120 (5)	0.674 (3)
Letter*	0.691 (2)	0.827 (1)	0.667 (3)	0.527 (5)	0.638 (4)
Lymphography	0.743 (3)	0.739 (4)	0.830 (1)	0.549 (5)	0.746 (2)
Mushrooms*	0.990 (1.5)	0.482 (5)	0.941 (3)	0.857 (4)	0.990 (1.5)
OptDigits*	0.867 (3)	0.098 (1)	0.915 (2)	0.986 (1)	0.784 (4)
Satimage*	0.821 (3)	0.872 (2)	0.815 (4)	0.885 (1)	0.778 (5)
SpamBase*	0.893 (2)	0.824 (4)	0.902 (1)	0.739 (5)	0.885 (3)
Splice*	0.799 (2)	0.655 (4)	0.925 (1)	0.517 (5)	0.755 (3)
Tic-tac-toe	0.845 (1)	0.731 (2)	0.693 (4)	0.653 (5)	0.704 (3)
Vehicle	0.741 (1)	0.701 (2)	0.591 (5)	0.663 (3)	0.619 (4)
Vowel	0.799 (2)	0.994 (1)	0.603 (4)	0.269 (5)	0.621 (3)
Wine	0.949 (4)	0.955 (2)	0.989 (1)	0.770 (5)	0.954 (3)
Yeast	0.555 (3)	0.505 (4)	0.569 (1)	0.312 (5)	0.556 (2)
Zoo	0.928 (2.5)	0.928 (2.5)	0.945 (1)	0.419 (5)	0.897 (4)
average rank	2.100	3.250	2.200	4.333	3.117

Table 2: Computation of the rankings for the five algorithms considered in the study over 30 data sets, based on test accuracy by using ten-fold cross validation

i	hypothesis	$z = (R_0 - R_i)/SE$	p	α_{NM}	α_{HM}	α_{SH}
1	C4.5 vs. Kernel	5.471	$4.487 \cdot 10^{-8}$	0.005	0.005	0.005
2	NaiveBayes vs. Kernel	5.226	$1.736 \cdot 10^{-7}$	0.005	0.0055	0.0083
3	Kernel vs. CN2	2.98	0.0029	0.005	0.0063	0.0083
4	C4.5 vs. 1NN	2.817	0.0048	0.005	0.0071	0.0083
5	1NN vs. Kernel	2.654	0.008	0.005	0.0083	0.0083
6	1NN vs. NaiveBayes	2.572	0.0101	0.005	0.01	0.0125
7	C4.5 vs. CN2	2.49	0.0128	0.005	0.0125	0.0125
8	NaiveBayes vs. CN2	2.245	0.0247	0.005	0.0167	0.0167
9	1NN vs. CN2	0.327	0.744	0.005	0.025	0.025
10	C4.5 vs. NaiveBayes	0.245	0.8065	0.005	0.05	0.05

Table 3: Family of hypotheses ordered by p-value and adjusting of α by Nemenyi (NM), Holm (HM) and Shaffer (SH) procedures, considering an initial $\alpha = 0.05$

Size 1	Size 2	Size 3	Size 4	Size ≥ 6
(12)	(12,34)	(12,13,23)	(12,13,23,45)	(12,13,14,15,23,24,25,34,35,45)
(13)	(13,24)	(12,14,24)	(12,14,24,35)	(12,13,14,23,24,34)
(23)	(14,23)	(13,14,34)	(12,34,35,45)	(12,13,15,23,25,35)
(14)	(12,35)	(23,24,34)	(13,14,25,34)	(12,14,15,24,25,45)
(24)	**(13,25)**	(12,15,25)	(13,15,24,35)	(13,14,15,34,35,45)
(34)	(15,23)	(13,15,35)	(13,24,25,45)	(23,24,25,34,35,45)
(15)	(12,45)	(23,25,35)	(14,15,23,45)	
(25)	(13,45)	(14,15,45)	(14,23,25,35)	
(35)	(23,45)	(24,25,45)	(15,23,24,34)	
(45)	(14,25)	(34,35,45)		
	(15,24)			
	(14,35)			
	(24,35)			

Table 4: Exhaustive sets obtained for the case study. Those belonging to the *Acceptance* set (A) are typed in bold.

In Demšar (2006), we can find a discussion about the power of Hochberg's and Hommel's procedures with respect to Holm's one. They reject more hypothesis than Holm's, but the differences are in practice rather small (Shaffer, 1995). The most powerful procedures detailed in this paper, Shaffer's and Bergmann-Hommel's, work following the same method of Holm's procedure, so it is possible to hybridize them with other types of step up procedures, such as Hochberg's, Hommel's and Rom's methods. When we apply these methods by using the logical relationships among hypothesis in a static way, they do not control the family-wise error (Hochberg and Rom, 1995). In opposite, when applying these methods by detecting dynamical relationships, they control the family-wise error. In Hochberg and Rom (1995), several extensions were given in this way. Furthermore, a small improvement of power in the Bergmann-Hommel procedure described here can be achieved when using Simes conjecture (Simes, 1986) in the obtaining of A set (see Hommel and Bernhard, 1999, for more details).

3. Adjusted P-Values

The smallest level of significance that results in the rejection of the null hypothesis, the p-value, is a useful and interesting datum for many consumers of statistical analysis. A p-value provides information about whether a statistical hypothesis test is significant or not, and it also indicates something about "how significant" the result is: The smaller the p-value, the stronger the evidence against the null hypothesis. Most important, it does this without committing to a particular level of significance.

When a p-value is within a multiple comparison, as in the example in Table 3, it reflects the probability error of a certain comparison, but it does not take into account the remaining comparisons belonging to the family. One way to solve this problem is to report APVs which take into account that multiple tests are conducted. An APV can be compared directly with any chosen significance level α. In this paper, we encourage the use of APVs due to the fact that they provide more information in a statistical analysis.

In the following, we will explain how to compute the APVs depending on the post-hoc procedure used in the analysis, following the indications given in Wright (1992) and Hommel and Bernhard (1999). We also include the post-hoc tests explained in Demšar (2006) and other for comparisons with a control classifier. The notation used in the computation of the APVs is the following:

- Indexes i and j correspond each one to a concrete comparison or hypothesis in the family of hypotheses, according to an incremental order by their p-values. Index i always refers to the hypothesis in question whose APV is being computed and index j refers to another hypothesis in the family.
- p_j is the p-value obtained for the j-th hypothesis.
- k is the number of classifiers being compared.
- m is the number of possible comparisons in an all pairwise comparisons design; that is, $m = \frac{k \cdot (k-1)}{2}$.
- t_j is the maximum number of hypotheses which can be true given that any $(j-1)$ hypotheses are false (see the description of Shaffer's static procedure in Section 2.1).

The procedures of p-value adjustment can be classified into:

- one-step.
  - Bonferroni APV_i: $min\{v;1\}$, where $v=(k-1)p_i$.
  - Nemenyi APV_i: $min\{v;1\}$, where $v=m\cdot p_i$.
- step-up.
  - Hochberg APV_i: $max\{(k-j)p_j : (k-1)\geq j\geq i\}$.
  - Hommel APV_i: see algorithm at Figure 3.
- step-down.
  - Holm APV_i (using a control classifier): $min\{v;1\}$, where $v=max\{(k-j)p_j : 1\leq j\leq i\}$.
  - Nemenyi APV_i: $min\{v;1\}$, where $v=m\cdot p_i$.
  - Holm APV_i (using it in all pairwise comparisons): $min\{v;1\}$, where $v=max\{(m-j+1)p_j : 1\leq j\leq i\}$.
  - Shaffer static APV_i: $min\{v;1\}$, where $v=max\{t_j p_j : 1\leq j\leq i\}$.
  - Bergmann-Hommel APV_i: $min\{v;1\}$, where $v=max\{|I|\cdot min\{p_j, j\in I\} : I\ exhaustive, i\in I\}$.

1. Set $APV_i=p_i$ for all i.
2. For each $j=k-1,k-2,...,2$ (in that order)
   3. Let $B=\emptyset$.
   4. For each i, $i>(k-1-j)$
      5. Compute value $c_i=(j\cdot p_i)/(j+i-k+1)$.
      6. $B=B\cup c_i$.
   7. End for
   8. Find the smallest c_i value in B; call it c_{min}.
   9. If $APV_i<c_{min}$, then $APV_i=c_{min}$.
   10. For each i, $i\leq(k-1-j)$
      11. Let $c_i=min(c_{min},j\cdot p_i)$.
      12. If $APV_i<c_i$, then $APV_i=c_i$.
   13. End for

Figure 3: Algorithm for calculating APVs based on Hommel's procedure

Table 5 shows the results in the final form of APVs for the example considered in this section. As we can see, this example is suitable for observing the difference of power among the test procedures. Also, this table can provide information about the state of retainment or rejection of any hypothesis, comparing its associated APV with the level of significance previously fixed.

i	hypothesis	p_i	APV_{NM}	APV_{HM}	APV_{SH}	APV_{BH}
1	C4.5 vs .Kernel	$4.487 \cdot 10^{-8}$	$4.487 \cdot 10^{-7}$	$4.487 \cdot 10^{-7}$	$4.487 \cdot 10^{-7}$	$4.487 \cdot 10^{-7}$
2	NaiveBayes vs .Kernel	$1.736 \cdot 10^{-7}$	$1.736 \cdot 10^{-6}$	$1.563 \cdot 10^{-6}$	$1.042 \cdot 10^{-6}$	$1.042 \cdot 10^{-6}$
3	Kernel vs .CN2	0.0029	0.0288	0.023	0.0173	0.0115
4	C4.5 vs .1NN	0.0048	0.0485	0.0339	0.0291	0.0291
5	1NN vs .Kernel	0.008	0.0796	0.0478	0.0478	0.0319
6	1NN vs .NaiveBayes	0.0101	0.1011	0.0506	0.0478	0.0319
7	C4.5 vs .CN2	0.0128	0.1276	0.0511	0.0511	0.0383
8	NaiveBayes vs .CN2	0.0247	0.2474	0.0742	0.0742	0.0383
9	1NN vs .CN2	0.744	1.0	1.0	1.0	1.0
10	C4.5 vs .NaiveBayes	0.8065	1.0	1.0	1.0	1.0

Table 5: APVs obtained in the example by Nemenyi (NM), Holm (HM), Shaffer's static (SH) and Bergmann-Hommel's dynamic (BH)

4. Experimental Framework

In this section, we want to determine the power and behavior of the studied procedures through the experiments in which we repeatedly compared the classifiers on sets of ten randomly chosen data sets, recording the number of equivalence hypothesis rejected and APVs. We follow a similar method used in Demšar (2006).

The classifiers used are the same as in the case study of the previous subsection: C4.5 with minimum number of item-sets per leaf equal to 2 and confidence level fitted for optimal accuracy and pruning strategy, naive Bayesian learner with continuous attributes discretized using Fayyad and Irani (1993) discretization, classic 1-Nearest-Neighbor classifier with Euclidean distance, CN2 with Fayyad-Irani's discretizer, star size = 5 and 95% of examples to cover and Kernel classifier with $sigmaKernel = 0.01$, which is the inverse value of the variance that represents the radius of neighborhood. All classifiers are available in KEEL software (Alcalá-Fdez et al., 2008).[5]

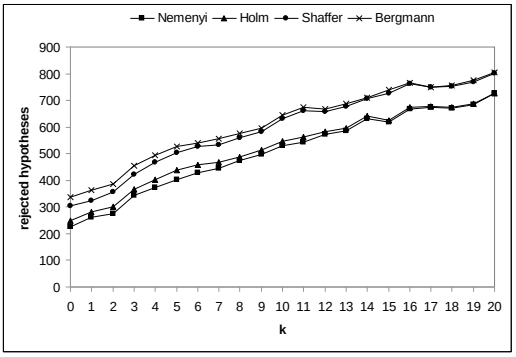
For performing this study, we have compiled a sample of fifty data sets from the UCI machine learning repository (Asuncion and Newman, 2007), all of them valid for a classification task.[6] We measured the performance of each classifier by means of accuracy in test by using ten-fold cross validation. As Demšar did, when comparing two classifiers, samples of ten data sets were randomly selected so that the probability for the data set i being chosen was proportional to $1/(1+e^{-kd_i})$, where d_i is the (positive or negative) difference in the classification accuracies on that data set and k is the bias through which we can regulate the differences between the classifiers. With $k = 0$, the selection is purely random and as k is being higher, the selected data sets are favorable to a particular classifier.

In comparisons of multiple classifiers, samples of data sets have to be selected with the probabilities computed from the differences in accuracy of two classifiers. We have chosen C4.5 and 1-NN, due to the fact that we have found significant differences between them in the study conducted before (Section 2.2) which involved thirty data sets. Note that the repeated comparisons done here only involve ten data sets each time, so the rejection of equivalence of two classifiers is more difficult at the beginning of the process.

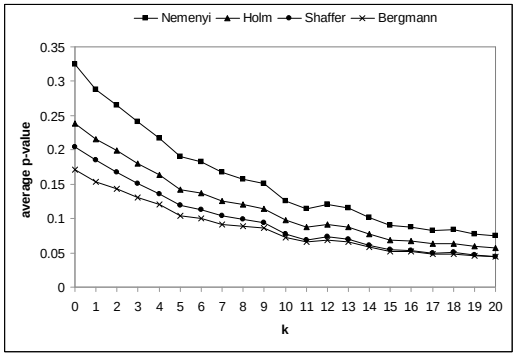
5. It is also available at `http://www.keel.es`.

6. The data sets used are: abalone, adult, australian, autos, balance, bands, breast, bupa, car, cleveland, dermatology, ecoli, flare, german, glass, haberman, hayes-roth, heart, iris, led7digit, letter, lymphography, magic, monks, mushrooms, newthyroid, nursery, optdigits, pageblocks, penbased, pima, ring, satimage, segment, shuttle, spambase, splice, tae, thyroid, tic-tac-toe, twonorm, vehicle, vowel, wine, wisconsin, yeast, zoo.

Figure 4 shows the results of this study considering the pairwise comparison between C4.5 and 1-NN. It gives an approximation of the power of the statistical procedures considered in this paper. Figure 4(a) reflects the number of times they rejected the equivalence of C4.5 and 1-NN. Obviously, the Bergmann-Hommel procedure is the most powerful, followed by Shaffer's static procedure. The graphic also informs us about the use of logically related hypothesis, given that the procedures that use this information have a bias towards the same point and those which do not use this information, tend to a lower point than the first. When the selection of data sets is purely random ($k = 0$), the benefit of using the Bergmann-Hommel procedure is appreciable. Figure 4(b) shows the average APV of the same comparison of classifiers. As we can see, the Nemenyi procedure is too conservative in comparison with the remaining procedures. Again, the benefit of using more sophisticated testing procedures is easily noticeable.



(a) Number of hypotheses rejected in pairwise comparisons



(b) Average APV in pairwise comparisons

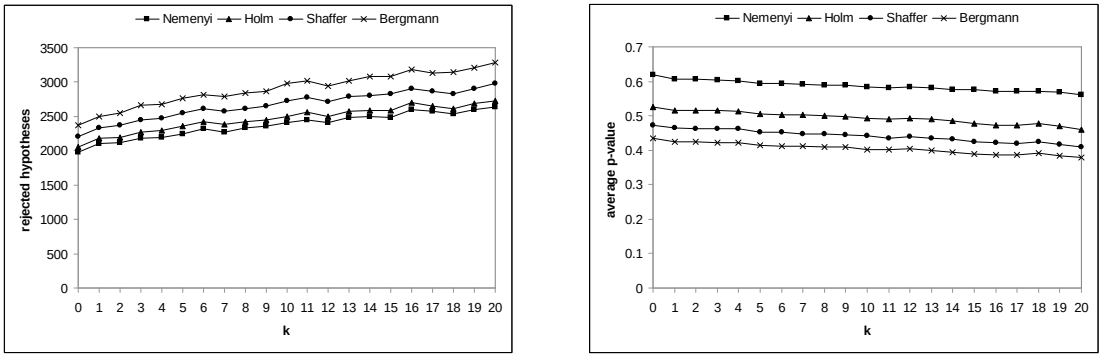
Figure 4: C4.5 vs. 1-NN

Figure 5 shows the results of this study considering all possible pairwise comparisons in the set of classifiers. It helps us to compare the overall behavior of the four testing procedures. Figure 5(a) presents the number of times they rejected any comparison belonging to the family. Although it could seem that the selection of data sets determined by the difference of accuracy between two classifiers may not influence on the overall comparison, the graphic shows us that it occurs. Furthermore, the lines drawn follow a parallel behavior, indicating us the relation and magnitude of power among the four procedures. In Figure 5(b) we illustrate the average APV for all the comparisons of classifiers. We can notice that the conservatism of the Nemenyi test is obvious with respect to the rest of procedures. The benefit of using a more advanced testing procedure is similar with respect to the following less-powerful procedure, except for Holm's procedure.

Finally, our recommendation on the usage of a certain procedure depends on the results obtained in this paper and in our experience in understanding and implementing them:

- We do not recommend the use of Nemenyi's test, because it is a very conservative procedure and many of the obvious differences may not be detected.

- When we use a considerable number of data sets with regards to number of classifiers, we could proceed with the Holm procedure.



(a) Total number of hypotheses rejected

(b) Average APV in all comparisons

Figure 5: All comparisons

- However, conducting the Shaffer static procedure means a not very significant increase of the difficulty with respect to the Holm procedure. Moreover, the benefit of using information about logically related hypothesis is noticeable, thus we strongly encourage the use of this procedure.

- Bergmann-Hommel's procedure is the best performing one, but it is also the most difficult to understand and computationally expensive. We recommend its usage when the situation requires so (i.e., the differences among the classifiers compared are not very significant), given that the results it obtains are as valid as using other testing procedure.

5. Conclusions

The present paper is an extension of Demšar (2006). Demšar does not deal in depth with some topics related to multiple comparisons involving all the algorithms and computations of adjusted p-values.

In this paper, we describe other advanced testing procedures for conducting all pairwise comparisons in a multiple comparisons analysis: Shaffer's static and Bergmann-Hommel's procedures. The advantage that they obtain is produced due to the incorporation of more information about the hypotheses to be tested: in $n \times n$ comparisons, a logical relationship among them exists. As a general rule, the Bergmann-Hommel procedure is the most powerful one but it requires intensive computation in comparisons involving numerous classifiers. The second one, Shaffer's procedure, can be used instead of Bergmann-Hommel's in these cases. Moreover, we present the methods for obtaining the adjusted p-values, which are valid p-values associated to each comparison useful to be compared with any level of significance without restrictions and they also provide more information. We have illustrated them with a case study and we have checked that the new described methods are more powerful than the classical ones, Nemenyi's and Holm's procedures.

Acknowledgments

This research has been supported by the project TIN2005-08386-C05-01. S. García holds a FPU scholarship from Spanish Ministry of Education and Science. The present paper was submitted as a regular paper in the JMLR journal. After the review process, the action editor Dale Schuurmans encourages us to submit the paper to the special topic on Multiple Simultaneous Hypothesis Testing. We are very grateful to the anonymous reviewers and both action editors who managed this paper for their valuable suggestions and comments to improve its quality.

Appendix A. Source Code of the Procedures

The source code, written in JAVA, that implements all the procedures described in this paper, is available at `http://sci2s.ugr.es/keel/multipleTest.zip`. The program allows the input of data in CSV format and obtains as output a LaTeX document.

References

J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, and F. Herrera. KEEL: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing. doi: 10.1007/s00500-008-0323-y*, 2008. In press.

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Anaylisis and Machine Intelligence*, 29(1):173–180, 2007.

G. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In P. Bauer, G. Hommel, and E. Sonnemann, editors, *Multiple Hypotheses Testing*, pages 100–115. Springer, Berlin, 1988.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

S. Esmeir and S. Markovitch. Anytime learning of decision trees. *Journal of Machine Learning Research*, 8:891–933, 2007.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1029. Morgan-Kaufmann, 1993.

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

N. García-Pedrajas and C. Fyfe. Immune network based ensembles. *Neurocomputing*, 70(7-9): 1155–1166, 2007.

Y. Hochberg. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75: 800–802, 1988.

Y. Hochberg and D. Rom. Extensions of multiple testing procedures based on Simes' test. *Journal of Statistical Planning and Inference*, 48:141–152, 1995.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70, 1979.

G. Hommel. A stagewise rejective multiple test procedure. *Biometrika*, 75:383–386, 1988.

G. Hommel and G. Bernhard. A rapid algorithm and a computer program for multiple test procedures using procedures using logical structures of hypotheses. *Computer Methods and Programs in Biomedicine*, 43:213–216, 1994.

G. Hommel and G. Bernhard. Bonferroni procedures for logically related hypotheses. *Journal of Statistical Planning and Inference*, 82:119–128, 1999.

R. L. Iman and J. M. Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics*, pages 571–595, 1980.

C. Marrocco, R. P. W. Duin, and F. Tortorella. Maximizing the area under the ROC curve by pairwise feature combination. *Pattern Recognition*, 41:1961–1974, 2008.

G. J. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley Series in Probability and Mathematical Statistics, 2004.

J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, 6:783–816, 2005.

P. B. Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, 1963.

M. Núñez, R. Fidalgo, and R. Morales. Learning in environments with unknown dynamics: Towards more robust concept learners. *Journal of Machine Learning Research*, 8:2595–2628, 2007.

A. B. Owen. Infinitely imbalanced logistic regression. *Journal of Machine Learning Research*, 8: 761–773, 2007.

J. R. Quinlan. *Programs for Machine Learning*. Morgan Kauffman, 1993.

D. M. Rom. A sequentially rejective test procedure based on a modified bonferroni inequality. *Biometrika*, 77:663–665, 1990.

J.P. Shaffer. Modified sequentially rejective multiple test procedures. *Journal of the American Statistical Association*, 81(395):826–831, 1986.

J.P. Shaffer. Multiple hypothesis testing. *Annual Review of Psychology*, 46:561–584, 1995.

D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures.* Chapman & Hall/CRC, 2003.

R.J. Simes. An improved Bonferroni procedure for multiple tests of significance. *Biometrika*, 73: 751–754, 1986.

P. H. Westfall and S. S. Young. *Resampling-Based Multiple Testing: Examples and Methods for p-value Adjustment*. John Wiley and Sons, 2004.

S. P. Wright. Adjusted p-values for simultaneous inference. *Biometrics*, 48:1005–1013, 1992.

Y. Yang, G. Webb, K. Korb, and K. M. Ting. Classifying under computational resource constraints: anytime classification using probabilistic estimators. *Machine Learning*, 69:35–53, 2007a.

Y. Yang, G. I. Webb, J. Cerquides, K. B. Korb, J. Boughton, and K. M. Ting. To select or to weigh: A comparative study of linear combination schemes for superparent-one-dependence estimators. *IEEE Transcations on Knowledge and Data Engineering*, 19(12):1652–1665, 2007b.

J. H. Zar. *Biostatistical Analysis*. Prentice Hall, 1999.

Journal of Machine Learning Research 9 (2008) 2695-2698 Submitted 5/08; Revised 9/08; Published 12/08

JNCC2: The Java Implementation Of Naive Credal Classifier 2

Giorgio Corani GIORGIO@IDSIA.CH
Marco Zaffalon ZAFFALON@IDSIA.CH
IDSIA
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
CH-6928 Manno (Lugano), Switzerland

Editor: Mikio Braun

Abstract

JNCC2 implements the *naive credal classifier 2* (NCC2). This is an extension of naive Bayes to imprecise probabilities that aims at delivering robust classifications also when dealing with small or incomplete data sets. Robustness is achieved by delivering set-valued classifications (that is, returning multiple classes) on the instances for which (i) the learning set is not informative enough to smooth the effect of choice of the prior density or (ii) the uncertainty arising from missing data prevents the reliable indication of a single class. JNCC2 is released under the GNU GPL license.

Keywords: imprecise probabilities, missing data, naive Bayes, naive credal classifier 2, Java

1. Introduction

JNCC2 is the Java implementation of *naive credal classifier 2* (NCC2) (Corani and Zaffalon, 2008). NCC2 extends naive Bayes (NBC) to *imprecise probabilities* (Walley, 1991) in order to deliver reliable classifications even on small or incomplete data sets.

A problem of NBC is that, on small data sets, it may return *prior-dependent* classifications, that is, it might identify a different class as the most probable one, depending on the prior density adopted to infer the classifier. In some cases this can lead NBC to issue fragile predictions. To deal with this problem, NCC2 specifies a set of prior densities, referred to as *prior credal set*; the credal set is then turned into a set of posteriors via element-wise application of Bayes' rule. Eventually, NCC2 returns the classes that are *non-dominated* with respect to the set of posterior densities (class c_1 dominates class c_2 if the probability of c_1 is larger than the probability of c_2 for *all* the posteriors). When faced with an instance that would be classified in a prior-dependent way by naive Bayes, NCC2 will detect multiple non-dominated classes and will then return multiple classes; this is an *indeterminate classification*.

As for missing data, NCC2 assumes that the missingness process (MP) which generates missing data can be either *MAR* (that is, missing at random), or unknown; in the latter case, it is referred to as non-MAR. As MAR missing data can be safely ignored (Little and Rubin, 1987), NCC2 ignores them. On the other hand, NCC2 deals *conservatively* with non-MAR missing data, that is, it considers all the possible replacements for non-MAR missing data. NCC2 can handle mixed situations where some features are subject to a MAR MP and some others to a non-MAR MP; moreover, the list of features subject to the MAR and to the unknown MP can be different between training and test set. The conservative treatment of non-MAR missing data generates additional

indeterminacy of NCC2, as a way to preserve reliability despite the information hidden by missing values and by the fact that the MP is unknown.

NCC2 can hence be seen as separating "easy" instances, over which it returns a single class, from "hard" instances, over which it returns an indeterminate classification. Experimental evaluations have shown that the accuracy of naive Bayes sharply drops on the hard instances, while on the same instances NCC2 remains reliable thanks to the indeterminate classifications.

Programming language: Java. *Developer*: Giorgio Corani (IDSIA, Switzerland). *Open source license*: GNU GPL. *Website*: `www.idsia.ch/~giorgio/jncc2.html`. *Software required*: Java Runtime Environment 5.0 or higher. *Operating system*: OS independent. *User interface*: command-line.

Figure 1: Essential information about JNCC2.

The zip file downloadable from the JNCC2 website contains executables, sources, examples, user manual and tutorial. A GUI version of the software will be released in the near future and will be published on the same website.

2. Indicators of Performance

The performance of NBC is measured by the *accuracy*, that is, the percentage of correct classifications.

The performance of NCC2 is measured by several indicators: *determinacy*: the percentage of classifications having as output a unique class; *single accuracy*: the accuracy of NCC2 when it is determinate; *indeterminate output size*: the average number of classes returned when NCC2 is indeterminate; *set-accuracy*: the percentage of indeterminate classifications that contain the actual class.

To assess the effectiveness of the approach based on imprecise probabilities, the accuracy of naive Bayes is moreover measured separately on the instances recognized as hard and easy by NCC2. If NCC2 is effective at separating easy from hard instances, a significant difference will be found between the two measures.

Moreover, JNCC2 computes the confusion matrices of NBC and NCC2; in case of NCC2 the confusion matrix refers to the determinate classifications only.

3. Some Implementation Details

JNCC2 loads data from ARFF files; this is a plain text format, originally developed for WEKA (Witten and Frank, 2005). A large number of ARFF data sets, including the data sets from the UCI repository, is available from the address `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`.

As a pre-processing step, JNCC2 discretizes all the numerical features, using the supervised discretization algorithm of Fayyad and Irani (1993). The discretization intervals are computed on the training set, and then applied unchanged on the test set.

NCC2 is implemented exploiting the computationally efficient procedure described in (Corani and Zaffalon, 2008, Appendix A).

Algorithm 1 Pseudocode for validation via testing file.

```
validateTestFile()

  /*loads training and test file; reads list of non-Mar features; discretizes features*/
  parseArffFile();
  parseArffTestingFile();
  parseNonMar();
  discretizeNumFeatures();

  /*learns and validates NBC*/
  nbc = new NaiveBayes(trainingSet);
  nbc.classifyInstances(testSet);

  /*learns and validates NCC2; the list of non-Mar features in training and testing is required*/
  ncc2 = new NaiveCredalClassifier2(trainingSet, nonMarTraining, nonMarTesting);
  ncc2.classifyInstances(testingSet);

  /*writes output files*/
  writePerfIndicators();
  writePredictions();
```

JNCC2 can perform three kinds of experiments: training and testing, cross-validation, and classification of instances of the test set whose class is unknown. The pseudo code of the experiment with training and testing is described by Algorithm 1.

4. Examples

To run the following examples, move to the directory `examples/completeData`, generated under the JNCC2 directory after unzipping the package. To perform a training and testing experiment, type for instance:

"`java jncc20.Jncc . iris.training.arff iris.testing.arff`".

As a consequence, JNCC2 will load the training and test set, discretize the numerical features, learn both NBC and NCC2, and use them to predict the instances of the test set. Then it will write the performance measures and the predictions to file. Similar experiments can be performed also with the glass and contact-lenses data sets, provided in the same directory.

To run a cross-validation experiment, type for instance:

"`java jncc20.Jncc . iris.training.arff cv`".

JNCC2 will perform 10 runs of 10-folds stratified cross-validation, that is, 100 training/test experiments. JNCC2 will report the performance indicators to file, together with their observed standard deviations, but it will not write the predictions. (As a side remark, if one wants to run cross-validation, there is no need of splitting the original data set into a training and a testing file, as it is has been done in this directory.)

The directory `examples/missingData` contains two examples of data sets containing missing data; a look at the provided files `NonMar.txt` should make it clear how to declare the non-MAR features.

The directory `examples/unkClasses` contains two examples in which the class of the instances of the testing set is not available. For the iris data set, the experiment is for instance started as follows:

```
"java jncc20.Jncc .  iris.training.arff iris.testingUnkClasses.arff
                                unknownclasses".
```

Acknowledgments

We are grateful to all the reviewers for their valuable comments. Work for this paper has been partially supported by the Swiss NSF grants 200021-113820/1 and 200020-116674/1, and by the Hasler Foundation (Hasler Stiftung) grant 2233.

References

G. Corani and M. Zaffalon. Learning reliable classifiers from small or incomplete data sets: the naive credal classifier 2. *Journal of Machine Learning Research*, 9:581–621, 2008.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, San Francisco, CA, 1993. Morgan Kaufmann.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. Wiley, New York, 1987.

P. Walley. *Statistical Reasoning With Imprecise Probabilities*. Chapman and Hall, New York, 1991.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (2nd Edition)*. Morgan Kaufmann, 2005.

Journal of Machine Learning Research 9 (2008) 2699-2731 Submitted 5/08; Published 12/08

Learning Bounded Treewidth Bayesian Networks

Gal Elidan GALEL@HUJI.AC.IL
Department of Statistics
Hebrew University
Jerusalem, 91905, Israel

Stephen Gould SGOULD@STANFORD.EDU
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA

Editor: David Maxwell Chickering

Abstract

With the increased availability of data for complex domains, it is desirable to learn Bayesian network structures that are sufficiently expressive for generalization while at the same time allow for tractable inference. While the method of thin junction trees can, in principle, be used for this purpose, its fully greedy nature makes it prone to overfitting, particularly when data is scarce. In this work we present a novel method for learning Bayesian networks of bounded treewidth that employs global structure modifications and that is polynomial both in the size of the graph and the treewidth bound. At the heart of our method is a dynamic triangulation that we update in a way that facilitates the addition of chain structures that increase the bound on the model's treewidth by at most one. We demonstrate the effectiveness of our "treewidth-friendly" method on several real-life data sets and show that it is superior to the greedy approach as soon as the bound on the treewidth is nontrivial. Importantly, we also show that by making use of global operators, we are able to achieve better generalization even when learning Bayesian networks of unbounded treewidth.

Keywords: Bayesian networks, structure learning, model selection, bounded treewidth

1. Introduction

Recent years have seen a surge of readily available data for complex and varied domains. Accordingly, increased attention has been directed towards the automatic learning of large scale probabilistic graphical models (Pearl, 1988), and in particular to the learning of the graph structure of a Bayesian network. With the goal of making predictions or providing probabilistic explanations, it is desirable to learn models that generalize well and at the same time have low inference complexity or a small treewidth (Robertson and Seymour, 1987).

Chow and Liu (1968) showed that the optimal Markov or Bayesian network can be learned efficiently when the underlying structure of the network is constrained to be a tree. Learning the structure of general Bayesian networks, however, is computationally difficult (Dagum and Luby, 1993), as is the learning of simpler structures such as poly-trees (Dasgupta, 1999) or even unconstrained chains (Meek, 2001). Several works try to generalize the work of Chow and Liu (1968) either by making assumptions about the generating distribution (e.g., Narasimhan and Bilmes, 2003; Abbeel et al., 2006), by searching for a local maxima of a mixture of trees model (Meila and Jordan, 2000), or by providing an approximate method that is polynomial in the size of the graph but

exponential in the treewidth bound (e.g., Karger and Srebro, 2001; Chechetka and Guestrin, 2008). In the context of general Bayesian networks, Bach and Jordan (2002) propose a local greedy approach that upper bounds the treewidth of the model at each step. Because evaluating the bound on the treewidth of a graph is super-exponential in the treewidth (Bodlaender, 1996), their approach relies on heuristic techniques for producing tree-decompositions (clique trees) of the model at hand, and uses that decomposition as an upper bound on the true treewidth of the model. This approach, like standard structure search, does not provide guarantees on the performance of the model, but is appealing in its ability to efficiently learn Bayesian networks with an arbitrary treewidth bound.

While tree-decomposition heuristics such as the one employed by Bach and Jordan (2002) are efficient and useful on average, there are two concerns when using such a heuristic in a fully greedy manner. First, even the best of heuristics exhibits some variance in the treewidth estimate (see, for example, Koster et al., 2001) and thus a single edge modification can result in a jump in the treewidth estimate despite the fact that adding a single edge to the network can increase the true treewidth by at most one. More importantly, most structure learning scores (e.g., BIC, MDL, BDe, BGe) tend to learn spurious edges that result in overfitting when the number of samples is relatively small, a phenomenon that is made worse by a fully greedy approach. Intuitively, to generalize well, we want to learn bounded treewidth Bayesian networks where structure modifications are globally beneficial (contribute to the score in many regions of the network).

In this work we propose a novel approach for efficiently learning Bayesian networks of bounded treewidth that addresses these concerns. At the heart of our method is the idea of dynamically updating a valid moralized triangulation of our model in a particular way, and using that triangulation to upper bound the model's treewidth. Briefly, we use a novel triangulation procedure that is treewidth-friendly: the treewidth of the triangulated graph is guaranteed to increase by at most one when an edge is added to the Bayesian network. Building on the single edge triangulation, we are also able to characterize sets of edges that *jointly* increase the treewidth of the triangulation by at most one. We make use of this characterization of treewidth-friendly edge sets in a dynamic programming approach that learns the optimal treewidth-friendly chain with respect to a node ordering. Finally, we learn a bounded treewidth Bayesian network by iteratively augmenting the model with such chains.

Importantly, instead of local edge modifications, our method progresses by making use of chain structure operators that are more globally beneficial, leading to greater robustness and improving our ability to generalize. At the same time, we are able to *guarantee* that the bound on the model's treewidth grows by at most one at each iteration. Thus, our method resembles the global nature of the method of Chow and Liu (1968) more closely than the thin junction tree approach of Bach and Jordan (2002), while being applicable in practice to any desired treewidth.

We evaluate our method on several challenging real-life data sets and show that our method is able to learn richer models that generalize better on test data than a greedy variant for a range of treewidth bounds. Importantly, we show that even when models with unbounded treewidth are learned, by employing global structure modification operators, we are better able to cope with the problem of local maxima in the search and learn models that generalize better.

The rest of the paper is organized as follows. After briefly discussing background material in Section 2, we provide a high-level overview of our approach in Section 3. In Section 4 we present our treewidth-friendly triangulation procedure in detail, followed by a multiple edge update discussion in Section 5. In Section 6 we show how to learn a treewidth-friendly chain given a node ordering and in Section 7 we propose a practical node ordering that is motivated by the properties of

our triangulation procedure. We evaluate the merits of our method in Section 8 and conclude with a discussion in Section 9.

2. Background

In this section we provide a basic review of Bayesian Networks as well as introduce the graph theoretic concepts of treewidth-decompositions and treewidth.

2.1 Bayesian Networks

Consider a finite set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of random variables. A *Bayesian network* (Pearl, 1988) is an annotated directed acyclic graph that encodes a joint probability distribution over $\mathcal{X}$. Formally, a Bayesian network over $\mathcal{X}$ is a pair $B = \langle \mathcal{G}, \Theta \rangle$. The first component, $\mathcal{G} = (V, E)$, is a directed acyclic graph whose vertices V correspond to the random variables in $\mathcal{X}$. The edges E in the graph represent direct dependencies between the variables. The graph $\mathcal{G}$ represents independence properties that are assumed to hold in the underlying distribution: each X_i is independent of its non-descendants given its parents $\mathbf{Pa}_i \subset \mathcal{X}$ denoted by $(X_i \perp \mathit{NonDescendants}_i \mid \mathbf{Pa}_i)$. The second component, Θ, represents the set of parameters that quantify the network. Each node is annotated with a *conditional probability distribution* $P(X_i \mid \mathbf{Pa}_i)$, representing the conditional probability of the node X_i given its parents in $\mathcal{G}$, defined by the parameters $\Theta_{X_i|\mathbf{Pa}_i}$. A Bayesian network defines a unique joint probability distribution over $\mathcal{X}$ given by

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid \mathbf{Pa}_i).$$

A *topological ordering* O_T of variables with respect to a Bayesian network structure is an ordering where each variable appears before all of its descendants in the network.

Given a Bayesian network model, we are interested in the task of probabilistic inference, or evaluating queries of the form $P_B(Y \mid Z)$ where Y and Z are arbitrary subsets of $\mathcal{X}$. This task is, in general, NP-hard (Cooper, 1990), except when $\mathcal{G}$ is tree structured. The actual complexity of inference in a Bayesian network (whether by variable elimination, by belief propagation in a clique tree, or by cut-set conditioning on the graph) is proportional to its *treewidth* (Robertson and Seymour, 1987) which, roughly speaking, measures how closely the network resembles a tree (see Section 2.2 for more details).

Given a network structure $\mathcal{G}$, the problem of learning a Bayesian network can be stated as follows: given a training set $\mathcal{D} = \{x[1], \ldots, x[M]\}$ of instances of $X \subseteq \mathcal{X}$, we want to learn parameters for the network. In the *Maximum Likelihood* setting we want to find the parameter values θ that maximize the log-likelihood function

$$\log P(\mathcal{D} \mid \mathcal{G}, \theta) = \sum_{m} \log P(x[m] \mid \mathcal{G}, \theta).$$

This function can be equivalently (up to a multiplicative constant) written as $\boldsymbol{E}_{\hat{P}}[\log P(X \mid \mathcal{G}, \theta)]$ where $\hat{P}$ is the empirical distribution in $\mathcal{D}$. When all instances in $\mathcal{D}$ are complete (that is, each training instance assigns values to all of the variables), estimating the *maximum likelihood* parameters can be done efficiently using a closed form solution for many choices of conditional probability distributions (for more details see Heckerman, 1998).

Learning the structure of a network poses additional challenges as the number of possible structures is super-exponential in the number of variables and the task is, in general, NP-hard (Chickering, 1996; Dasgupta, 1999; Meek, 2001). In practice, structure learning is typically done using a local search procedure, which examines local structure changes that are easily evaluated (add, delete or reverse an edge). This search is usually guided by a scoring function such as the MDL principle based score (Lam and Bacchus, 1994) or the *Bayesian score* (BDe) (Heckerman et al., 1995). Both scores penalize the likelihood of the data to limit the model complexity. An important characteristic of these scoring functions is that when the data instances are complete the score is *decomposable*. More precisely, a decomposable score can be rewritten as the sum

$$Score(\mathcal{G}:\mathcal{D}) = \sum_i \text{FamScore}_{X_i}(\mathbf{Pa}_i \ : \ \mathcal{D}).$$

where $\text{FamScore}_{X_i}(\mathbf{Pa}_i \ : \ \mathcal{D})$ is the *local* contribution of X_i to the total network score. This term depends only on values of X_i and $\mathbf{Pa}_{X_i}$ in the training instances.

Chow and Liu (1968) showed that maximum likelihood trees can be learned efficiently via a maximum spanning tree whose edge weights correspond to the empirical information between the two variables corresponding to the edge's endpoints. Their result can be easily generalized for any decomposable score.

2.2 Tree-Decompositions and Treewidth

The notions of tree-decompositions (or clique trees) and treewidth were introduced by Robertson and Seymour (1987).[1]

Definition 2.1: A tree-decomposition of an undirected graph $\mathcal{H} = (V,E)$ is a pair $(\{C_i\}_{i\in\mathcal{T}}, \mathcal{T})$ with $\{C_i\}_{i\in\mathcal{T}}$ a family of subsets of V, one for each node of $\mathcal{T}$, and $\mathcal{T}$ a tree such that

- $\bigcup_{i\in\mathcal{T}} C_i = V$.
- for all edges $(v,w) \in E$ there exists an $i \in \mathcal{T}$ with $v \in C_i$ and $w \in C_i$.
- for all $i,j,k \in \mathcal{T}$: if j is on the (unique) path from i to k in $\mathcal{T}$, then $C_i \cap C_k \subseteq C_j$.

∎

The treewidth of a tree-decomposition $(\{C_i\}_{i\in\mathcal{T}}, \mathcal{T})$ is defined to be $\max_{i\in\mathcal{T}} |C_i| - 1$. The treewidth $TW(\mathcal{H})$ of an undirected graph $\mathcal{H}$ is the minimum treewidth over all possible tree-decompositions of $\mathcal{H}$. An equivalent notion of treewidth can be phrased in terms of a graph that is a triangulation of $\mathcal{H}$.

Definition 2.2: An induced path $\mathcal{P} = p_1 — p_2 \ldots p_L$ in an undirected graph $\mathcal{H}$ is a path such that for every non-adjacent $p_i, p_j \in \mathcal{P}$ there is no edge $(p_i — p_j)$ in $\mathcal{H}$. An induced (non-chordal) cycle is an induced path whose endpoints are the same vertex. ∎

Definition 2.3: A triangulated or chordal graph is an undirected graph that has no induced cycles. Equivalently, it is an undirected graph in which every cycle of length greater than three contains a chord. ∎

1. The properties defining a tree-decomposition are equivalent to the corresponding *family preserving* and *running intersection* properties of clique trees introduced by Lauritzen and Spiegelhalter (1988) at around the same time.

It can be easily shown (Robertson and Seymour, 1987) that the treewidth of a given triangulated graph is the size of the maximal clique of the graph minus one. The treewidth of an undirected graph $\mathcal{H}$ is then equivalently the minimum treewidth over all possible triangulations of $\mathcal{H}$.

For the underlying directed acyclic graph of a Bayesian network, the treewidth can be characterized via a triangulation of the moralized graph.

Definition 2.4: A moralized graph $\mathcal{M}$ of a directed acyclic graph $\mathcal{G}$ is an undirected graph that includes an edge $(i—j)$ for every edge $(i \rightarrow j)$ in $\mathcal{G}$ and an edge $(p—q)$ for every pair of edges $(p \rightarrow i), (q \rightarrow i)$ in $\mathcal{G}$. ∎

The treewidth of a Bayesian network graph $\mathcal{G}$ is defined as the treewidth of its moralized graph $\mathcal{M}$, and corresponds to the complexity of inference in the model. It follows that the maximal clique of *any* moralized triangulation of $\mathcal{G}$ is an upper bound on the treewidth of the model, and thus its inference complexity.[2]

3. Learning Bounded Treewidth Bayesian Networks: Overview

Our goal is to develop an efficient algorithm for learning Bayesian networks with an arbitrary treewidth bound. As learning the optimal such network is NP-hard (Dagum and Luby, 1993), it is important to note the properties that we would like our algorithm to have. First, we would like our algorithm to be *provably* polynomial in the number of variables *and* in the desired treewidth. Thus, we cannot rely on methods such as that of Bodlaender (1996) to verify the boundedness of our network as they are super-exponential in the treewidth and are practical only for small treewidths. Second, we want to learn networks that are non-trivial. That is, we want to ensure that we do not quickly get stuck in local maxima due to the heuristic employed for bounding the treewidth of our model. Third, similar to the method of Chow and Liu (1968), we want to employ global structure operators that are optimal in some sense. In this section we present a brief high-level overview of our algorithm. In the next sections we provide detailed description of the different components along with proof of correctness and running time guarantees.

At the heart of our method is the idea of using a dynamically maintained moralized triangulated graph to upper bound the treewidth of the current Bayesian network. When an edge is added to the Bayesian network we update this (moralized) triangulated graph in a particular manner that is not only guaranteed to produce a valid triangulation, but that is also treewidth-friendly. That is, our update is guaranteed to increase the size of the maximal clique of the triangulated graph, and hence the treewidth bound, by at most one. As we will see, the correctness of our treewidth-friendly edge update as well as the fact that we can carry it out efficiently will both directly rely on the dynamic nature of our method. We discuss our edge update procedure in detail in Section 4.

An important property of our edge update is that we can characterize the parts of the network that are "contaminated" by the update by using the notion of blocks (bi-connected components) in the triangulated graph. This allows us to define sets of edges that are *jointly* treewidth-friendly. That is, these edge sets are guaranteed to increase the treewidth of the triangulated graph by at most one when all edges in the set are added to the Bayesian network structure. We discuss multiple edge updates in Section 5.

2. It also follows that the size of a family (a node and its parents) provides a lower bound on the treewidth, although we will not make use of this property in our work.

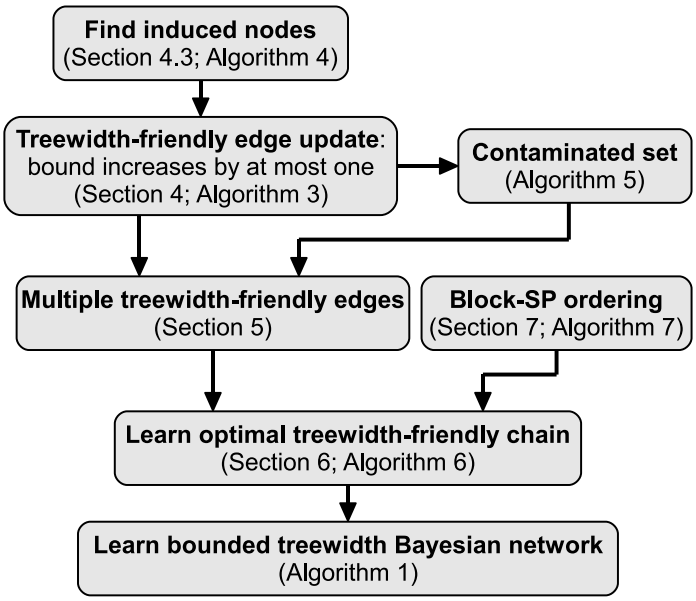


Figure 1: The building blocks of our method for learning Bayesian networks of bounded treewidth and how they depend on each other.

Building on the characterization of treewidth-friendly sets, we propose a dynamic programming approach for efficiently learning the optimal treewidth-friendly chain with respect to a node ordering. We present this procedure in Section 6. To encourage chains that are rich in structure (have many edges), in Section 7 we propose a block shortest-path node ordering that is motivated by the properties of our triangulation procedure.

Finally, we learn Bayesian networks with bounded treewidth by starting with a Chow-Liu tree (Chow and Liu, 1968) and iteratively applying a global structure modification operator where the current structure is augmented with a treewidth-friendly chain that is optimal with respect to the ordering chosen. Appealingly, as each global modification can increase our estimate of the treewidth by at most one, if our bound on the treewidth is K, at least K such chains will be added before we even face the problem of local maxima. In practice, as some chains do not increase the treewidth, many more such chains are added for a given maximum treewidth bound. Figure 1 illustrates the relationship between the different components of our approach.

Algorithm (1) shows pseudo-code of our method. Briefly, Line 4 initializes our model with a Chow and Liu (1968) tree; Line 8 produces a node ordering given the model at hand; Line 9 finds the optimal chain with respect to that ordering; and Line 10 augments the current model with the new edges. We then use our treewidth-friendly edge update procedure to perform the moralization and triangulation on $\mathcal{M}^+$ for each edge added to the Bayesian network $\mathcal{G}$ (Line 12). Once the maximal clique size reaches the treewidth bound K, we continue to add edges greedily until no more edges can be added without increasing the treewidth (Line 16).

Theorem 3.1: Given a treewidth bound K, Algorithm (1) runs in time polynomial in the number of variables and K.

Algorithm 1: Learning A Bayesian Network with Bounded Treewidth

Input : $\mathcal{D}$ // training set
K // maximum treewidth
Output: $\mathcal{G}$ // a graph structure with treewidth at most K

$\mathcal{G} \leftarrow$ maximum scoring spanning tree
$\mathcal{M}^+ \leftarrow$ undirected skeleton of $\mathcal{G}$
$k \leftarrow 1$
while $k < K$ *and positive scoring edges exist* **do**
  $O \leftarrow$ node ordering given $\mathcal{G}$ and $\mathcal{M}^+$ // Algorithm (7)
  $\mathcal{C} \leftarrow$ maximum scoring chain with respect to O // Algorithm (6)
  $\mathcal{G} \leftarrow \mathcal{G} \cup \mathcal{C}$
  foreach $(i \rightarrow j) \in \mathcal{C}$ **do**
    $\mathcal{M}^+ \leftarrow$ EdgeUpdate$(\mathcal{M}^+, (i \rightarrow j))$ // Algorithm (3)
  end foreach
  $k \leftarrow$ maximal clique size of $\mathcal{M}^+$
end
Greedily add edges to $\mathcal{G}$ that do not increase treewidth beyond K
return $\mathcal{G}$

We will prove this result gradually using the developments of the next sections. Note that we will show that our method is guaranteed to be polynomial both in the size of the graph *and* the treewidth bound. Thus, like the greedy thin junction tree approach of Bach and Jordan (2002), it can be used to learn a Bayesian networks given an arbitrary treewidth bound. It is also important to note that, as in the case of the thin junction tree method, the above result is only useful if the actual Bayesian network learned is expressive enough to be useful for generalization. As we will demonstrate in Section 8, by making use of global treewidth-friendly updates, our method indeed improves on the greedy approach and learns models that are rich and useful in practice.

4. Treewidth-Friendly Edge Update

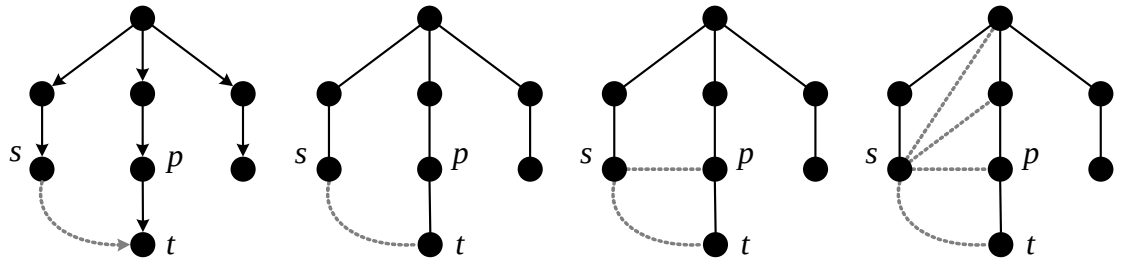
In this section we consider the basic building block of our method: the manner in which we update the triangulated graph when a single edge is added to the Bayesian network structure. Throughout this section we will build on the dynamic nature of our method and make use of the valid moralized triangulation graph that was constructed before adding an edge $(s \rightarrow t)$ to the Bayesian network structure. We will start by augmenting it with $(s$—$t)$ and any edges required for moralization. We will then triangulate the graph in a treewidth-friendly way, increasing the size of the maximal clique in the triangulated graph by at most one. For clarity of exposition, we start with a simple variant of our triangulation procedure in Section 4.1 and refine it in Section 4.2.

4.1 Single-source Triangulation

To gain intuition into how the dynamic nature of our update is useful, we use the notion of induced paths or paths with no shortcuts (see Section 2), and make explicit the following obvious fact.

Algorithm 2: SingleSourceEdgeUpdate: Update of $\mathcal{M}^+$ when adding $(s \rightarrow t)$ to $\mathcal{G}$

1 **Input** : $\mathcal{M}^+$ // triangulated moralized graph of $\mathcal{G}$
2 $(s \rightarrow t)$ // edge to be added to $\mathcal{G}$
3 **Output**: ${\mathcal{M}^+}_{(s \rightarrow t)}$ // a triangulated moralized graph of $\mathcal{G} \cup (s \rightarrow t)$
4 ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow \mathcal{M}^+ \cup (s—t)$
5 **foreach** $p \in \mathbf{Pa}_t$ **do**
6 ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow {\mathcal{M}^+}_{(s \rightarrow t)} \cup (s—p)$ // moralization
7 **end foreach**
8 **foreach** *node v on an induced path between s and* $t \cup \mathbf{Pa}_t$ *in* $\mathcal{M}^+$ **do**
9 ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow {\mathcal{M}^+}_{(s \rightarrow t)} \cup (s—v)$
10 **end foreach**
11 **return** ${\mathcal{M}^+}_{(s \rightarrow t)}$



(a) Bayesian network $\mathcal{G} \cup (s \rightarrow t)$

(b) Addition of $(s—t)$ to $\mathcal{M}^+$ (line 4)

(c) Addition of moralizing edges (lines 5-7)

(d) Addition of the triangulating edges (lines 8-10)

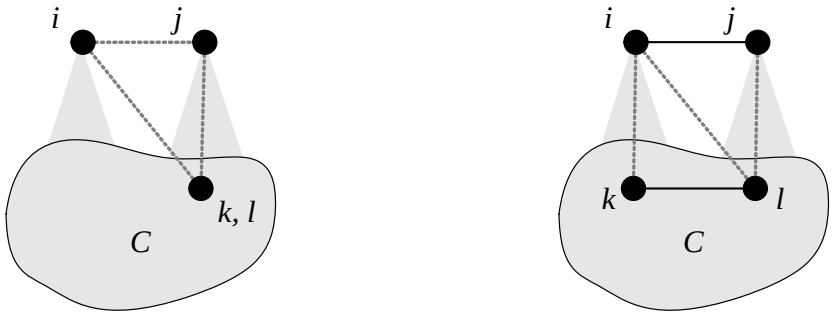
Figure 2: Example showing the application of the single-source triangulation procedure of Algorithm (2) to a simple graph. The treewidth of the original graph is one, while the graph augmented with $(s \rightarrow t)$ has a treewidth of two (maximal clique of size three).

Observation 4.1: Let $\mathcal{G}$ be a Bayesian network structure and let $\mathcal{M}^+$ be a moralized triangulation of $\mathcal{G}$. Let $\mathcal{M}_{(s \rightarrow t)}$ be $\mathcal{M}^+$ augmented with the edge $(s—t)$ and with the edges $(s—p)$ for every parent p of t in $\mathcal{G}$. Then, every non-chordal cycle in $\mathcal{M}_{(s \rightarrow t)}$ involves s and either t or a parent of t and an induced path between the two vertices.

Stated simply, if the graph was triangulated before the addition of $(s \rightarrow t)$ to the Bayesian network, then we only need to triangulate cycles created by the addition of the new edge or those forced by moralization. This observation immediately suggests the straight-forward *single-source triangulation* outlined in Algorithm (2): add an edge $(s—v)$ for every node v on an induced path between s and t or s and a parent p of t before the edge update. Figure 2 shows an application of the procedure to a simple graph. Clearly, this naive method results in a valid moralized triangulation of $\mathcal{G} \cup (s \rightarrow t)$. Surprisingly, we can also show that it is treewidth-friendly.

Theorem 4.2: The treewidth of the output graph ${\mathcal{M}^+}_{(s \rightarrow t)}$ of Algorithm (2) is greater than the treewidth of the input graph $\mathcal{M}^+$ by at most one.

Proof: Let C be the nodes in any maximal clique $\mathcal{M}^+$. We consider the minimal set of edges required to increase the size of C by more than one and show that this set cannot be a subset of the edges added by our single-source triangulation. In order for the clique to grow by more than one node, at least two nodes i and j not originally in C must become connected to all nodes in C. Since there exists at least one node $k \in C$ that is not adjacent to i and similarly there exists at least one node $l \in C$ not adjacent to j, both edges $(i—k)$ and $(j—l)$ are needed to form the larger clique. There are two possibilities illustrated below (the dotted edges are needed to increase the treewidth by two and all other edges between i, j and the current maximal clique are assumed to exist):

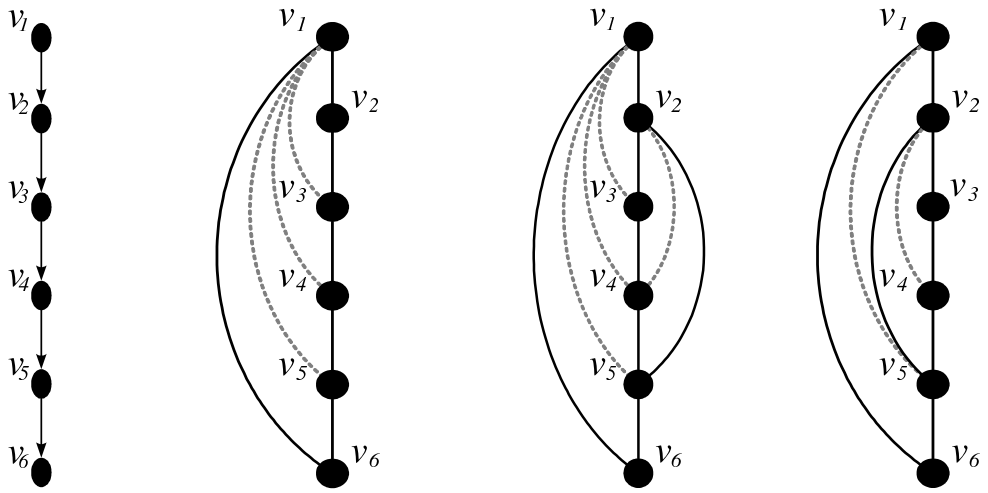


(a) $(i—j)$ does not exist (b) $(i—j)$ exists

- $(i—j)$ **does not exist (a)**. In this case k and l can be the same node but the missing edge $(i—j)$ is also required to form the larger clique.
- $(i—j)$ **exists (b)**. In this case k and l cannot be the same node or the original clique was not maximal since $C \cup i \cup j \setminus k$ would have formed a larger clique. Furthermore one of k or l must not be connected to both i and j otherwise $i—j—k—l—i$ forms a non-chordal cycle of length four contradicting our assumption that the original graph was triangulated. Thus, in this case either $(i—l)$ or $(j—k)$ are also required to form the larger clique.

In both scenarios, at least two nodes have two incident edges and the three edges needed cannot all be incident to a single vertex. Now consider the triangulation procedure. Since, by construction, all edges added in Algorithm (2) emanate from s, the above condition (requiring two nodes to have two incident edges and the three edges not all incident to a single vertex) is not met and the size of the maximal clique in the new graph cannot be larger than the size of the maximal clique in $\mathcal{M}^+$ by more than one. It follows that the treewidth of the moralized triangulated graph cannot increase by more than one. ∎

One problem with the proposed single-source triangulation, despite it being treewidth-friendly, is the fact that so many vertices are connected to the source node making the triangulations shallow (the length of the shortest path between any two nodes is small). While this is not a problem when considering a single edge update, it can have an undesirable effect on future edges and increases the chances of the formation of large cliques. As an example, Figure 3 shows a simple case where two successive single-source edge updates increase the treewidth by two while an alternative approach increases the treewidth by only one. In the next section, we present a refinement of the single-source triangulation that is motivated by this example.



(a) Chain network (b) Triangulation after $(v_1 \rightarrow v_6)$ is added (c) Triangulation after $(v_2 \rightarrow v_5)$ is added (d) Alternative triangulation

Figure 3: Example demonstrating that the single-source edge update of Algorithm (2) can be problematic for later edge additions. (a) shows a simple six nodes chain Bayesian network; (b) a single-source triangulation when $(v_1 \rightarrow v_6)$ is added to the network with a treewidth of two; (c) a single-source triangulation when in addition $(v_2 \rightarrow v_5)$ is added to the model with a treewidth of three; (d) an alternative triangulation to (b). This triangulation already includes the edge $(v_2$—$v_5)$ and the moralizing edge $(v_2$—$v_4)$ and thus is also a valid moralized triangulation after $(v_2 \rightarrow v_5)$ has been added, but has a treewidth of only two.

4.2 Alternating Cut-vertex Triangulation

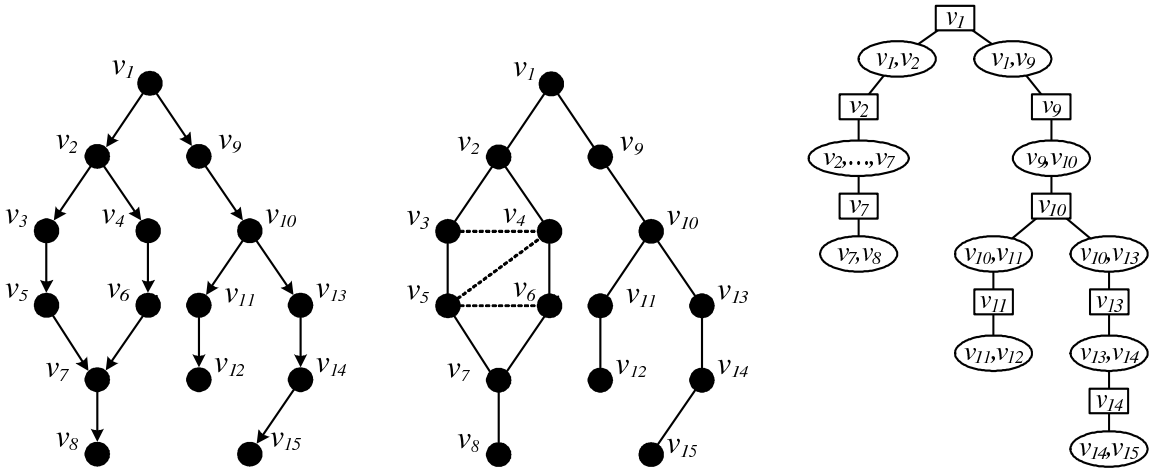
To refine the single-source triangulation discussed above with the goal of addressing the problem exemplified in Figure 3 we make use of the concepts of cut-vertices, blocks, and block trees (see, for example, Diestel, 2005).

Definition 4.3: A block, or biconnected component, of an undirected graph is a set of connected nodes that cannot be disconnected by the removal of a single vertex. By convention, if the edge $(u$—$v)$ is in the graph then u and v are in the same block. Vertices that separate (are in the intersection of) blocks are called cut-vertices. ∎

It follows directly from the definition that between every two nodes in a block (of size greater than two), there are at least two distinct paths, that is, a cycle. There are also no simple cycles involving nodes in different blocks.

Definition 4.4: A block tree $\mathcal{B}$ of an undirected graph $\mathcal{H}$ is a graph with nodes that correspond both to cut-vertices and to blocks of $\mathcal{H}$. The edges in the block tree connect any block node B_i with a cut-vertex node v_j if and only if $v_j \in B_i$ in $\mathcal{H}$. ∎

It can be easily shown that the above connectivity condition indeed forces a tree structure and that this tree is unique (see Figure 4 for an example). In addition, any path in $\mathcal{H}$ between two nodes



(a) Bayesian network, $\mathcal{G}$ (b) A possible triangulated graph, $\mathcal{M}^+$ (c) Unique block tree, $\mathcal{B}$

Figure 4: Example of a Bayesian network with a corresponding moralized triangulated graph and the unique block tree. Boxes in the block tree denote cut-vertices, ellipses denote blocks.

in different blocks passes through all the cut-vertices along the path between the blocks in $\mathcal{B}$. An important consequence that directly follows from the result of Dirac (1961) is that an undirected graph whose blocks are triangulated is overall triangulated.

We can now describe our improved treewidth-friendly triangulation outlined in Algorithm (3) and illustrated via an example in Figure 5. First, the triangulated graph is augmented by the edge $(s—t)$ and any edges needed for moralization (Figure 5(b) and (c)). Second, if s and t are not in the same block, a block level triangulation is carried out by starting from s and zig-zagging across the cut-vertices along the unique path between the blocks containing s and t and its parents in the block tree (Figure 5(d)). Next, within each block along the path (not containing s or t), a chord is added between the "entry" and "exit" cut-vertices along the *block path*, thereby short-circuiting any other *node path* through the block. In addition, within each such block we perform a single-source triangulation with respect to s' by adding an edge $(s'—v)$ between the first cut-vertex s' and any node v on an induced path between s' and the second cut-vertex t'. The block containing s is treated the same as other blocks on the path with the exception that the short-circuiting edge is added between s and the first cut-vertex along the path from s to t. For the block containing t and its parents, instead of adding a chord between the entry cut-vertex and t, we add chords directly from s to any node v (within the block) that is on an *induced path* between s and t (or parents of t) (Figure 5(e)). This is required to prevent moralization and triangulation edges from interacting in a way that will increase the treewidth by more than one (see Figure 5(f) for an example). If s and t happen to be in the same block, then we only triangulate the induced paths between s and t, that is, the last step outlined above. Finally, in the special case that s and t are in *disconnected* components of $\mathcal{G}$, the only edges added are those required for moralization.

We now show that this revised edge update is a valid triangulation procedure and that it is also treewidth-friendly. To do so we start with the following observations that are a direct consequence of the definition of a block and block tree.

Algorithm 3: EdgeUpdate: Update of $\mathcal{M}^+$ when adding $(s \rightarrow t)$ to $\mathcal{G}$

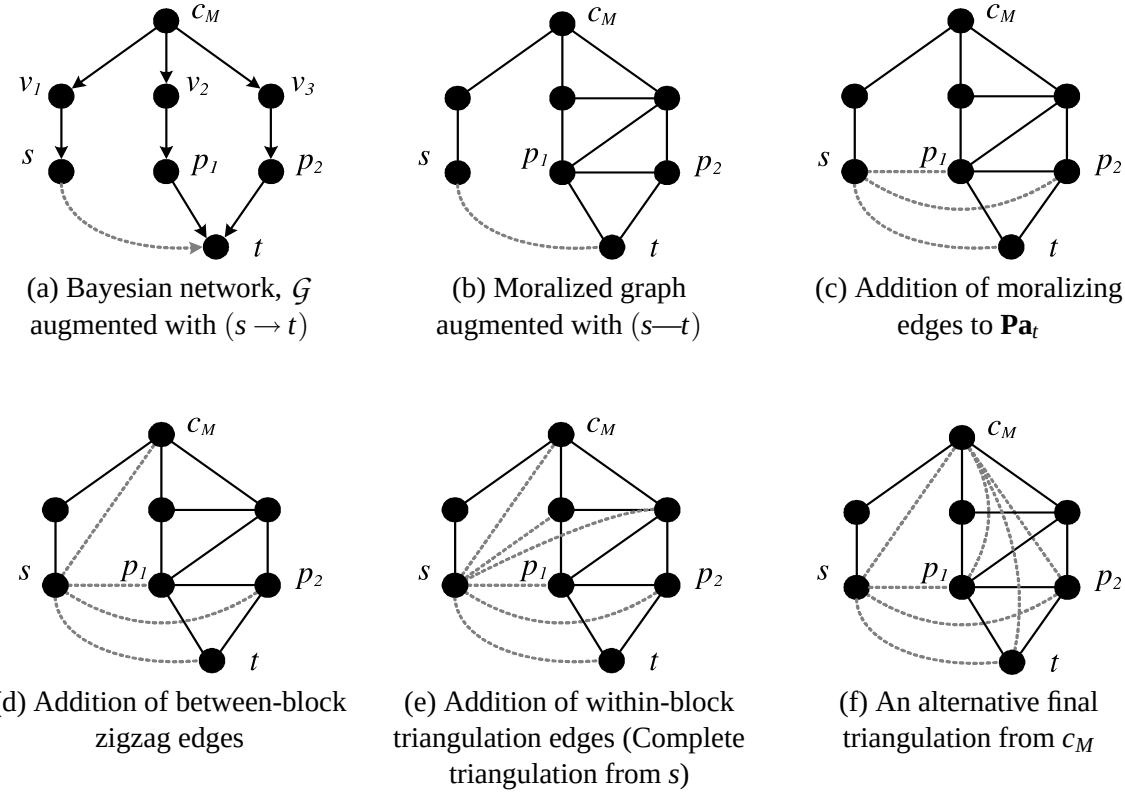
Input : $\mathcal{M}^+$ // triangulated moralized graph of $\mathcal{G}$
O // node ordering
$(s \rightarrow t)$ // edge to be added to $\mathcal{G}$
Output: ${\mathcal{M}^+}_{(s \rightarrow t)}$ // a triangulated moralized graph of $\mathcal{G} \cup (s \rightarrow t)$

$\mathcal{B} \leftarrow$ block tree of $\mathcal{M}^+$
${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow \mathcal{M}^+ \cup (s — t)$
foreach $p \in \mathbf{Pa}_t$ **do**
  ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow {\mathcal{M}^+}_{(s \rightarrow t)} \cup (s — p)$ // moralization
end foreach
// triangulate (cut-vertices) between blocks
$\mathcal{C} = \{c_1, \ldots, c_M\} \leftarrow$ sequence of cut-vertices on the path from s to $t \cup \mathbf{Pa}_t$ in block tree $\mathcal{B}$
Add $(s — c_M), (c_M — c_1), (c_1 — c_{M-1}), (c_{M-1} — c_2), \ldots$ to ${\mathcal{M}^+}_{(s \rightarrow t)}$
// triangulate nodes within blocks on path from s to $t \cup \mathbf{Pa}_t$
$\mathcal{E} \leftarrow \{(s — c_1), (c_1 — c_2), \ldots, (c_{M-1} — c_M)\}$
foreach *edge* $(s' — t') \in \mathcal{E}$ **do**
  ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow {\mathcal{M}^+}_{(s \rightarrow t)} \cup (s' — t')$
  foreach *node v on an induced path between s' and t' in the* **original** *block containing both* **do**
    ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow {\mathcal{M}^+}_{(s \rightarrow t)} \cup (s' — v)$
  end foreach
end foreach
// triangulate s with nodes in block containing $t \cup \mathbf{Pa}_t$
foreach *node v on an induced path between s and $t \cup \mathbf{Pa}_t$ in the* **new** *block containing them* **do**
  ${\mathcal{M}^+}_{(s \rightarrow t)} \leftarrow {\mathcal{M}^+}_{(s \rightarrow t)} \cup (s — v)$
end foreach
return ${\mathcal{M}^+}_{(s \rightarrow t)}$

Observation 4.5: (Family Block). Let u be a node in a Bayesian network $\mathcal{G}$ and let $\mathbf{Pa}_u$ be the set of parents of u. Then the block tree for any moralized triangulated graph $\mathcal{M}^+$ of $\mathcal{G}$ has a unique block containing $\{u, \mathbf{Pa}_u\}$.

Observation 4.6: (Path Nodes). Let $\mathcal{B} = (\{B_i\} \cup \{c_j\}, \mathcal{T})$ be the block tree of $\mathcal{M}^+$ with blocks $\{B_i\}$ and cut-vertices $\{c_j\}$. Let s and t be nodes in blocks B_s and B_t, respectively. If t is a cut-vertex then let B_t be the (unique) block that also contains $\mathbf{Pa}_t$. If s is a cut-vertex, then choose B_s to be the block containing s closest to B_t in $\mathcal{T}$. Then a node v is on a path from s to t or from s to a parent of t if and only if it is in a block that is on the unique path from B_s to B_t.

Figure 4(c) shows an example of a block tree for a small Bayesian network. Here, for example, selecting s to be the node v_6 and t to be the node v_{10} in $\mathcal{G}$, it is clear that all paths between s and t include only the vertices that are in blocks along the unique block path between B_s and B_t. Furthermore, every path between s and t passes through all the cut-vertices on this block path, that



(a) Bayesian network, $\mathcal{G}$ augmented with $(s \to t)$

(b) Moralized graph augmented with $(s—t)$

(c) Addition of moralizing edges to $\mathbf{Pa}_t$

(d) Addition of between-block zigzag edges

(e) Addition of within-block triangulation edges (Complete triangulation from s)

(f) An alternative final triangulation from c_M

Figure 5: Example showing our triangulation procedure (b)-(e) for s and t in different blocks. (The blocks are $\{s, v_1\}$, $\{v_1, c_M\}$, and $\{c_M, v_2, v_3, p_1, p_2, t\}$ with corresponding cut-vertices v_1 and c_M). The original graph has a treewidth of two, while the graph augmented with $(s \to t)$ has treewidth three (maximal clique of size four). An alternative triangulation (f), connecting c_M to t, however, would result in a clique of size five $\{s, c_M, p_1, p_2, t\}$.

is, $\{v_2, v_1, v_9\}$. We can now use these properties to show that our edge update procedure produces a valid triangulation.

Lemma 4.7: *If $\mathcal{M}^+$ is a valid moralized triangulation of the graph $\mathcal{G}$ then Algorithm (3) produces a moralized triangulation $\mathcal{M}^+{}_{(s \to t)}$ of the graph $\mathcal{G}_{(s \to t)} \equiv \mathcal{G} \cup (s \to t)$.*

Proof: Since $\mathcal{M}^+$ was triangulated, every cycle of length greater than or equal to four in $\mathcal{G}_{(s \to t)}$ is the result of the edge $(s—t)$ or one of the moralizing edges, together with an induced path (path with no shortcuts) between the endpoints of the edge. We consider three cases:

- s **and** t **are disconnected in** $\mathcal{M}^+$**:** There are no induced paths between s and t so the only edges required are those for moralization. These edges do not produce any induced cycles.

- s **and** t **are in the same block:** The edge $(s—t)$ does not create a new block and all simple cycles that involve both s and t must be within the block. Thus, by construction, the edges added in Line 16 triangulate all newly introduced induced cycles. If the parents of t are in the same block as s and t, the same reasoning holds for all induced paths between a parent p of

t and s. Otherwise, t is a cut-vertex between the block that contains its parents and the block that contains s. It follows that all paths (including induced ones) from a parent of t to s pass through t and the edges added for the s,t-block triangulate all newly created induced cycles that result from the moralizing edges.

- **s and t are not in the same block:** As noted in Observation 4.6, all paths in $\mathcal{M}^+$ from s to t or a parent of t pass through the unique cut-vertex path from the block containing s to the block containing t and its parents. The edges added in Line 14 short-circuit the in-going s' and out-going t' of each block creating a path containing only cut-vertices between s and t. Line 11 triangulates this path by forming cycles of length three containing s', t' and some other cut-vertex. The only induced cycles remaining are contained within blocks and contain the newly added edge $(s'—t')$ or involve the edge between s and the last cut-vertex $(s—c_M)$ and one of the edges between s and t or a parent of t. It follows that within-block triangulation with respect to s' and t' will shortcut the former induced cycles, and the edges added from s in Line 20 will shortcut the later induced cycles.

To complete the proof, we need to show that any edge added from s (or s') to an induced node v does not create new induced cycles. Any such induced cycle would have to include an induced path from the endpoints of the edge added and thus would have been a sub-path of some induced cycle that includes both s and v. This cycle would have already been triangulated by our procedure. ∎

Having shown that our update produces a valid triangulation, we now prove that our edge update is indeed treewidth-friendly and that it can increase the treewidth of the moralized triangulated graph by at most one.

Theorem 4.8: The treewidth of the output graph $\mathcal{M}^+{}_{(s\rightarrow t)}$ of Algorithm (3) is greater than the treewidth of the input graph $\mathcal{M}^+$ by at most one.

Proof: As shown in the proof of Theorem 4.2, the single-source triangulation within a block is guaranteed not to increase the maximal clique size by more than one. In addition, from the properties of blocks it follows directly that the inner block triangulation does not add edges that are incident to nodes outside of the block. It follows that all the inner block single-source triangulations independently effect disjoint cliques. Thus, the only way that the treewidth of the graph can further increase is via the zig-zag edges. Now consider two cliques in different blocks. Since our block level zig-zag triangulation only touches two cut-vertices in each block, it cannot join two cliques of size greater than two into a single larger one. In the simple case of two blocks with two nodes (a single edge) and that intersect at a single cut-vertex, a zig-zag edge can indeed increase the treewidth by one. In this case, however, there is no within-block triangulation and so the overall treewidth cannot increase by more than one. ∎

4.3 Finding Induced Nodes

We finish the description of our edge update (Algorithm (3)) by showing that it can be carried out efficiently. That is, we have to be able to efficiently find the vertices on *all* induced paths between two nodes in a graph. In general, this task is computationally difficult as there are potentially exponentially many such paths between any two nodes. To cope with this problem, we again make use of the dynamic nature of our method.

The idea is simple. As implied by Observation 4.1, any induced path between s' and t' in a triangulated graph will be part of an induced cycle if $(s'—t')$ is added to the graph. Furthermore,

Algorithm 4: InducedNodes: compute set of nodes on induced path between s' and t' in $\mathcal{M}^+$

```
1  Input  : M+    // moralized triangulated graph
2           s',t' // two nodes in M+
3  Output: I      // set of nodes on induced paths between s' and t'
4  H ← block (subgraph) of M+ ∪ (s'—t') containing s' and t'
5  I ← ∅
6  while edges being added do
       // maximum cardinality search
7      X ← all nodes in H except s'
8      Y ← {s'}
9      while X ≠ ∅ do
10         Find v ∈ X with maximum number of neighbors in Y
11         X ← X \ {v} and Y ← Y ∪ {v}   // remove from X, add to Y
12         if there exists u,w ∈ Y such that (u—w) ∉ H then
13             I ← I ∪ {u,v,w}
14             Add edges (s'—u), (s'—v) and (s'—w) to H
15             Restart maximum cardinality search
16         end
17     end
18 end
19 return I
```

after adding $(s'—t')$ to the graph, *every* cycle detected will involve an induced path between the two nodes. Using this observation, we can make use of the ability of the maximum cardinality search algorithm (Tarjan and Yannakakis, 1984) to iteratively detect non-chordal cycles.

The method is outlined in Algorithm (4). At each iteration we attempt to complete a maximum cardinality search starting from s' (Line 7 to Line 17). If the search fails, we add the node at which it failed, v, together with its non-adjacent neighboring nodes $\{u,w\}$ to the set of induced nodes and augment the graph with triangulating edges from s' to each of $\{u,v,w\}$. If the search completes then we have successfully triangulated the graph and hence found all induced nodes. Note that using the properties of blocks and cut-vertices, we only need to consider the subgraph that is the block created after the addition of $(s'—t')$ to the graph.

Lemma 4.9: (Induced Nodes). *Let $\mathcal{M}^+$ be a triangulated graph and let s' and t' be any two nodes in $\mathcal{M}^+$. Then Algorithm (4) efficiently returns all nodes on any induced path between s' and t' in $\mathcal{M}^+$, unless those nodes are connected directly to s'.*

Proof: During a maximum cardinality search, if the next node chosen v has two neighbors u and w that are not connected then the triplet $u—v—w$ is part of an induced cycle. As the graph was triangulated before adding the edge $(s'—t')$, all such cycles must contain s' and adding $(s'—v)$ obviously shortcuts such a cycle. This is also true for v and v' that are on the same induced cycle. It remains to show that the edges added do not create new induced cycle. Such an induced cycle would have to include the edge $(s'—v)$ as well as an induced path between s' and v. However, such

a path must have been part of another cycle where v was an induced node and hence would have been triangulated. ∎

Thus Algorithm (4) returns exactly the set of nodes on induced paths from s' to t' that s' needs to connect to in order to triangulate the graph $\mathcal{M}^+ \cup (s—t)$. The efficiency of our edge update procedure of Algorithm (3) follows immediately as all other operations are simple.

5. Multiple Edge Updates

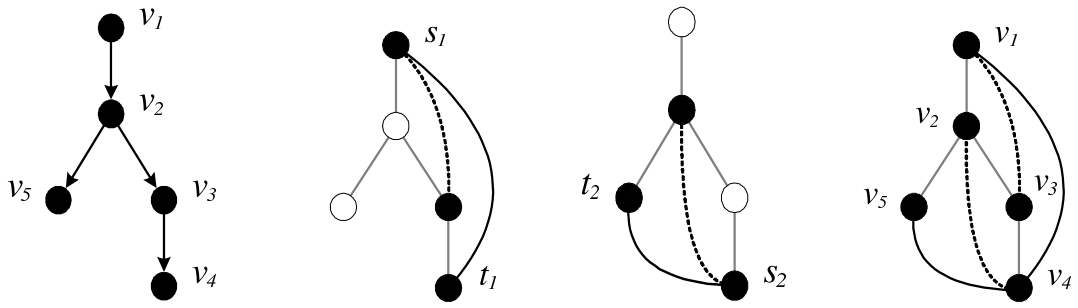
In this section we define the notion of a *contaminated set*, or the subset of nodes that are incident to edges added to $\mathcal{M}^+$ in Algorithm (3), and characterize sets of edges that are jointly guaranteed not to increase the treewidth of the triangulated graph by more than one. We begin by formally defining the terms *contaminate* and *contaminated set*.

Definition 5.1: We say that a node v is contaminated by the addition of the edge $(s \rightarrow t)$ to $\mathcal{G}$ if it is incident to an edge added to the moralized triangulated graph $\mathcal{M}^+$ by a call to Algorithm (3). The contaminated set for an edge $(s \rightarrow t)$ is the set of all nodes v that would be contaminated (with respect to $\mathcal{M}^+$) by adding $(s \rightarrow t)$ to $\mathcal{G}$, including s, t, and the parents of t. ∎

Figure 6 shows some examples of contaminated sets for different edge updates. Note that our definition of contaminated set only includes nodes that are incident to *new* edges added to $\mathcal{M}^+$ and, for example, excludes nodes that were already connected to s before $(s \rightarrow t)$ is added, such as the two nodes adjacent to s in Figure 6(b).

Using the separation properties of cut-vertices, one might be tempted to claim that if the contaminated sets of two edges overlap at most by a single cut-vertex then the two edges jointly increase the treewidth by at most one. This however, is not true in general as the following example shows.

Example 5.2: Consider the Bayesian network shown below in (a) and its triangulation (b) after $(v_1 \rightarrow v_4)$ is added, increasing the treewidth from one to two. (c) is the same for the case when $(v_4 \rightarrow v_5)$ is added to the network. Despite the fact that the contaminated sets (solid nodes) of two edge additions overlap only by the cut-vertex v_4, (d) shows that jointly adding the two edges to the graph results in a triangulated graph with a treewidth of three. ∎



(a) (b) (c) (d)

The problem in the above example lies in the overlap of *block paths* between the endpoints of the two edges, a property that we have to take into account while characterizing sets of treewidth-friendly edges.

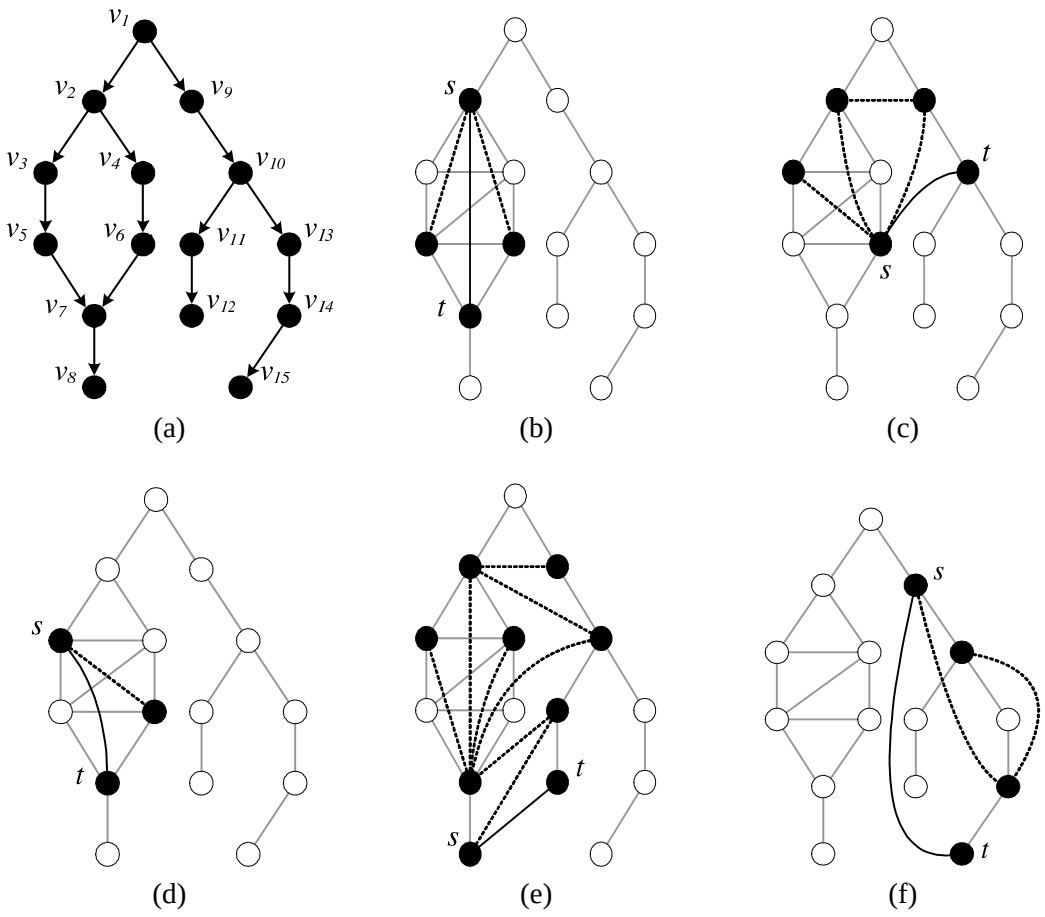


Figure 6: Some examples of contaminated sets (solid nodes) that are incident to edges added (dashed) by Algorithm (3) for different candidate edge additions $(s \rightarrow t)$ to the Bayesian network shown in (a). In (b), (c), (d), and (e) the treewidth is increased by one; In (f) the treewidth does not change.

Theorem 5.3: (Treewidth-friendly pair). Let $\mathcal{G}$ be a Bayesian network graph structure and $\mathcal{M}^+$ be its corresponding moralized triangulation. Let $(s \rightarrow t)$ and $(u \rightarrow v)$ be two distinct edges that are topologically consistent with $\mathcal{G}$. Then the addition of the edges to $\mathcal{G}$ does not increase the treewidth of $\mathcal{M}^+$ by more than one if *one* of the following conditions holds:

- the contaminated sets of $(s \rightarrow t)$ and $(u \rightarrow v)$ are disjoint.
- the endpoints of each of the two edges are not in the same block *and* the block paths between the endpoints of the two edges do not overlap *and* the contaminated sets of the two edge overlap at a single cut-vertex.

Proof: As in the proof of Algorithm (3) a maximal clique can grow by two nodes only if three undirected edges are added so that at least two nodes are incident to two of them. Obviously, this

Algorithm 5: ContaminatedSet: compute contaminated set for $(s \rightarrow t)$

Input : $\mathcal{G}$ // Bayesian network
$\mathcal{M}^+$ // moralized triangulated graph
$(s \rightarrow t)$ // candidate edge
Output: $\mathcal{C}_{s,t}$ // contaminated set for $(s \rightarrow t)$

$\mathcal{C}_{s,t} \leftarrow \{s,t\} \cup \{p \in \mathbf{Pa}_t \mid (s\text{—}p) \notin \mathcal{M}^+\}$
foreach *edge* $(s'\text{—}t') \in \mathcal{E}$ *in procedure Algorithm (3) with* $(s'\text{—}t') \notin \mathcal{M}^+$ **do**
  $I = \text{InducedNodes}(\mathcal{M}^+, \{s',t'\})$
  $\mathcal{C}_{s,t} \leftarrow \mathcal{C}_{s,t} \cup \{v \in I \mid (s'\text{—}v) \notin \mathcal{M}^+\}$
end foreach
$\mathcal{H} \leftarrow \{s\}$ and block containing $t \cup \mathbf{Pa}_t$
$\mathcal{H} \leftarrow \mathcal{H} \cup \{(s\text{—}p) \mid p \in \mathbf{Pa}_t\} \cup (s\text{—}c)$ where c is the cut-vertex closest to s in the block containing t
$I = \text{InducedNodes}(\mathcal{H}, \{s,t\})$
$\mathcal{C}_{s,t} \leftarrow \mathcal{C}_{s,t} \cup \{v \in I \mid (s'\text{—}v) \notin \mathcal{M}^+\}$
return $\mathcal{C}_{s,t}$

can only happen if the contamination sets of the two edge updates are not completely disjoint. Now, consider the case when the two sets overlap by a single cut-vertex. By construction all triangulating edges added are along the block path between the endpoints of each edge. Since the block paths of the two edge updates do not overlap there can not be an edge between a node in the contaminated set of $(s \rightarrow t)$ and the contaminated set of $(u \rightarrow v)$ (except for the single cut-vertex). But then no node from either contaminated set can become part of a clique involving nodes from the other contaminated set. Thus there are no two nodes that can be added to the same clique. It follows that the maximal clique size of $\mathcal{M}^+$, and hence the treewidth bound, cannot grow by more than one. ∎

The following result is an immediate consequence.

Corollary 5.4: (Treewidth-friendly set). *Let $\mathcal{G}$ be a Bayesian network graph structure and $\mathcal{M}^+$ be its corresponding moralized triangulation. If $\{(s_i \rightarrow t_i)\}$ is a set of edges so that every pair of edges satisfies the condition of Theorem 5.3 then adding all edges to $\mathcal{G}$ can increase the treewidth bound by at most one.*

The above result characterizes treewidth-friendly sets. In the search for such sets that are useful for generalization (see Section 6), we will need be able to efficiently compute the contaminated set of candidate edges. At the block level, adding an edge between s and t in $\mathcal{G}$ can only contaminate blocks between the block containing s and that containing t and its parents in the block tree for $\mathcal{M}^+$ (Observation 4.6). Furthermore, identifying the nodes that are incident to moralizing edges and edges that are part of the zigzag block level triangulation is easy. Finally, within a block, the contaminated set is easily computed using Algorithm (4) for finding the induced nodes between two vertices. Algorithm (5) outlines this procedure. Its correctness follows directly from the correctness of Algorithm (4) and the fact that it mirrors the edge update procedure of Algorithm (3).

6. Learning Optimal Treewidth-Friendly Chains

We now want to build on the results of the previous sections to facilitate the addition of global moves that are both optimal in some sense and are guaranteed to increase the treewidth by at most one. Specifically, we consider adding optimal chains that are consistent with some topological node ordering. On the surface, one might question the need for a specific node ordering altogether if chain global operators are to be used—given the result of Chow and Liu (1968), one might expect that learning the optimal chain with respect to *any* ordering can be carried out efficiently. However, Meek (2001) showed that learning such an optimal chain over a set of random variables is computationally difficult. Furthermore, conditioning on the current model, the problem of identifying the optimal chain is equivalent to learning the (unconditioned) optimal chain.[3] Thus, during any iteration of our algorithm, we cannot expect to find the overall optimal chain.

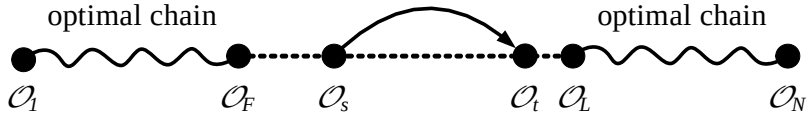
Instead, we commit to a single node ordering that is topologically consistent and learn the optimal chain *with respect to that order*. In this section we will complete the development of our algorithm and show how we can efficiently learn chains that are optimal with respect to any such ordering. In Section 7 we will also suggest a useful node ordering motivated by the characteristics of contaminated sets. We start by formally defining the chains that we will learn.

Definition 6.1: A treewidth-friendly chain $\mathcal{C}$ with respect to a node ordering O is a chain with respect to O such that the contamination conditions of Theorem 5.3 hold for the set of edges in $\mathcal{C}$. ∎

Given a treewidth-friendly chain $\mathcal{C}$ to be added for Bayesian network $\mathcal{G}$, we can apply the edge update of Algorithm (3) successively to every edge in $\mathcal{C}$ to produce a valid moralized triangulation of $\mathcal{G} \cup \mathcal{C}$. The result of Theorem 5.4 ensures that the resulting moralized triangulation will have treewidth at most one greater than the original moralized triangulation $\mathcal{M}^+$.

To find the optimal treewidth-friendly chain in polynomial time, we use a straightforward dynamic programming approach: the best treewidth-friendly chain that contains $(O_s \rightarrow O_t)$ is the concatenation of

- the best treewidth-friendly chain from the first node in the order O_1 to O_F, the first ordered node contaminated by the edge $(O_s \rightarrow O_t)$
- the edge $(O_s \rightarrow O_t)$
- the best treewidth-friendly chain starting from O_L, the last node contaminated by the edge $(O_s \rightarrow O_t)$, to the last node in the order, O_N.



We note that when the end nodes are not separating cut-vertices, we maintain a gap so that the contamination sets are disjoint and the conditions of Theorem 5.3 are met.

3. Consider, for example, the star-network where a single node acts as parent to all other nodes (with no other edges), then learning the optimal chain amounts to learning a chain over the $n-1$ children.

Formally, we define $C[i,j]$ as the optimal chain whose contamination starts at or after O_i and ends at or before O_j. To find the optimal treewidth-friendly chain with respect to a node ordering O for a Bayesian network with N variables, our goal is then to compute $C[1,N]$. Using the shorthand notation F to denote the first node ordered in the contamination set of $(s \rightarrow t)$ and L to denote the last ordered node in the contamination set, we can readily compute $C[1,N]$ via the following recursive update principle

$$C[i,j] = \max \begin{cases} \max_{s,t:F=i,L=j}(s \rightarrow t) & \text{no split} \\ \max_{k=i+1:j-1} C[i,k] \cup C[k,j] & \text{split} \\ \emptyset & \text{leave a gap} \end{cases}$$

where the maximization is with respect to the score (e.g., BIC) of the structures considered. In simple words, the maximum chain in any sub-sequence $[i,j]$ in the node ordering is the maximum of three alternatives: all edges whose contamination boundaries are exactly i and j (no split); all two chain combinations that are in the sub-sequence $[i,j]$ and are joined at some node $i < k < j$ (split); a gap between i and j in the case that there is no edge whose contamination is contained in this range and that increases the score.

Algorithm (6) outlines a simple backward recursion that computes $C[1,N]$. At each node, the algorithm maintains a list of the best partial chains evaluated so far that contaminates nodes up to, but not preceding, that node in the ordering. That is, the list of best partial chains is indexed by where the contamination boundary of each chain starts in the ordering. By recursing backwards from the last node, the algorithm is able to update this list by evaluating all candidate edges *terminating* at the current node. It follows that, once the algorithm iterates past a node t we have the optimal chain *starting* from that node. Thus, at the end of the recursion we are left with the optimal non-contaminating chain starting from the first node in the ordering.

The recursion starts at Line 7. If for node O_t the best chain starting from the succeeding node O_{t+1} is better than the best chain starting from O_t, we replace the best chain from O_t with the one from O_{t+1} simply leaving a gap in the chain (Line 8). Then, for every edge terminating at O_t, we find the first ordered node O_F and the last ordered node O_L that would be contaminated by adding that edge. If the score for the edge plus the score for the best partial non-contaminating chain from O_F is better than the current best partial chain from O_L then we replace the best chain from O_L with the one just found (Line 19).

With the ability to learn optimal chains with respect to a node ordering, we have completed the description of all the components of our algorithm for learning bounded treewidth Bayesian network outlined in Algorithm (1). Its efficiency is a direct consequence of our ability to learn treewidth-friendly chains in time that is polynomial both in the number of variables and in the treewidth at each iteration. For completeness we now restate and prove Theorem 3.1.

Theorem 3.1: Given a treewidth bound K, Algorithm (1) runs in time polynomial in the number of variables and K.

Proof: The initial Chow-Liu tree and its corresponding undirected skeleton can be obtained in polynomial time using a standard max-spanning-tree algorithm. The maximum scoring chain can be computed in polynomial time (using Algorithm (6)) at each iteration. As we proved, the same is true of the triangulation procedure of Algorithm (3). All other steps are trivial. Since the algorithm adds at least one edge per iteration it cannot loop for more than $K \cdot N$ iterations before exceeding a treewidth of K (where N is the number of variables). ∎

Algorithm 6: LearnChain: learn optimal non-contaminating chain with respect to topological node ordering

Input : $\mathcal{O}$ // topological node ordering
Output: $\mathcal{C}$ // non-contaminating chain

// initialize dynamic programming data
for $i = 1$ **to** $|\mathcal{O}| + 1$ **do**
  bestChain$[i] \leftarrow \emptyset$ // best chain from i-th node
  bestScore$[i] \leftarrow 0$ // best score from i-th node
end

// backward recursion
for $t = |\mathcal{O}|$ **down to** 1 **do**
  if (bestScore$[t+1] >$ bestScore$[t]$) **then**
    bestChain$[t] \leftarrow$ bestChain$[t+1]$
    bestScore$[t] \leftarrow$ bestScore$[t+1]$
  end
  for $s = 1$ **to** $t-1$ **do** // evaluate edges
    $\mathcal{V} \leftarrow$ contaminated set for candidate edge $(O_s \rightarrow O_t)$
    $f \leftarrow$ first ordered node in $\mathcal{V}$ // must be $\leq s$
    $l \leftarrow$ last ordered node in $\mathcal{V}$ // must be $\geq t$
    if bestChain$[l]$.last *and* $(O_s \rightarrow O_t)$ *do not satisfy the conditions of Theorem 5.3* **then**
      $l \leftarrow l+1$ // leave a gap
    end
    if ($\Delta Score(O_s \rightarrow O_t) +$ bestScore$[l] >$ bestScore$[f]$) **then**
      bestChain$[f] \leftarrow \{(O_s \rightarrow O_t)\} \cup$ bestChain$[l]$
      bestScore$[f] \leftarrow \Delta Score(O_s \rightarrow O_t) +$ bestScore$[l]$
    end
  end
end

// return optimal non-contaminating chain
return bestChain$[1]$

7. Block-Shortest-Path Ordering

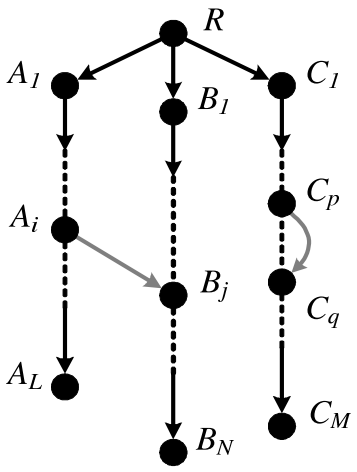
In the previous sections we presented an algorithm for learning bounded treewidth Bayesian networks given any topological ordering of the variables. In order to make the most of our method, we would like our ordering to facilitate rich structures that will have beneficial generalization properties. Toward that end, in this section we consider the practical matter of a concrete node ordering. We will present a block shortest-path (BSP) node ordering that is motivated by the specific properties of our triangulation method.[4]

4. We also considered several other strategies for ordering the variables. As none was better than the intuitive ordering described here, we only present results for our block-shortest-path ordering.

To make our node ordering concrete, since the contamination resulting from edges added within an existing block is limited to the block, we start by grouping together all nodes that are within a block (cut-vertices that appear in multiple blocks are included in the first block chosen). Our node ordering is then a topologically consistent ordering over the blocks combined with a topologically consistent ordering over the nodes within each block. We use topological consistency to facilitate as many edges as possible though this is not required by the theory (and, in particular, Theorem 5.3).

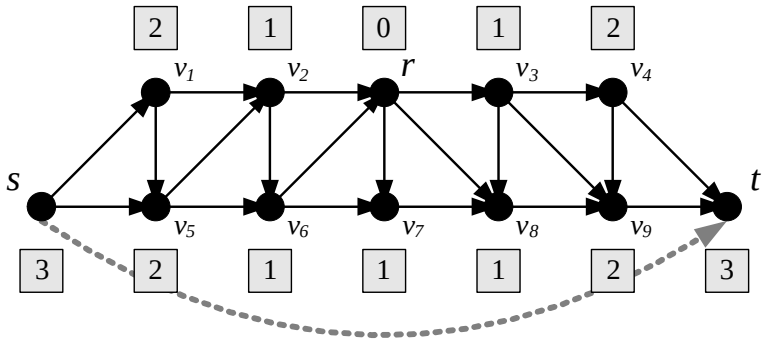
We now consider how to order interchangeable blocks by taking into account that our triangulation following an edge addition $(s \to t)$ only involves variables that are in blocks along the unique path between the block containing s and the block containing t and its parents. The following example motivates a natural choice for this ordering.

Example 7.1: Consider a Bayesian network with root node R and three branches: $R \to A_1 \to \ldots \to A_L$, $R \to B_1 \to \ldots \to B_N$, and $R \to C_1 \to \ldots \to C_M$. If we add an edge $A_i \to B_j$ to the network, then by the block contamination results, our triangulation procedure will touch (almost) every node on the path between A_i and B_j. This implies that we can not include additional edges of the type $B_k \to C_l$ in our chain since the block path from B_k to R overlaps with the block path from B_j to R. Note, however, that any edge $C_p \to C_{q>p}$ is still allowed to be added since its contaminated set does not overlap with that of $A_i \to B_j$. Now, consider the two obvious topological node orderings: $O^{\text{BFS}} = (R, A_1, B_1, C_1, A_2, \ldots)$ and $O^{\text{DFS}} = (R, A_1, \ldots, A_L, B_1, \ldots, B_N, C_1, \ldots)$. Only the DFS ordering, obtained by grouping the B_i's together, allows us to consider such edge combinations. ∎



Motivated by the above example to order interchangeable blocks, we use a block level depth-first ordering. The question now is whether a further characterization of the contaminated set can be provided in order to better order topologically interchangeable nodes within a block. To answer this question we consider the following example.

Example 7.2: Consider the Bayesian network shown below whose underlying undirected structure is a valid moralized triangulation and forms a single block. Numbers in the boxes indicate the (undirected) distance of each node from r, a property that we make use of below.



The single edge addition $(s \rightarrow t)$ will contaminate every node in the block (other than those already adjacent to it) since all nodes lie on induced paths between s and t. However other edge additions, such as $(v_3 \rightarrow t)$ have a much smaller contamination set: $\{v_3, t\}$. ∎

Based on the above example, one may think that no within-block ordering can improve the expected contamination of edges added, and that we may be forced to only add a single edge per block, making our method greedier than we would like. Fortunately, there is a straightforward way to characterize the within-block contamination set using the notion of shortest path length. Let $\mathcal{G}$ be a Bayesian network over variables $\mathcal{X}$. We denote by $d^M_{\min}(u,v)$ the minimum distance (shortest path) between nodes $u, v \in \mathcal{X}$ in $\mathcal{M}^+$. We note the following useful properties of $d^M_{\min}(\cdot,\cdot)$:

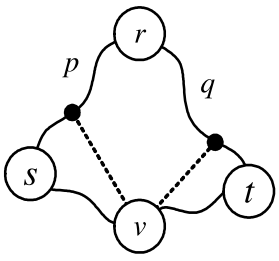
- $d^M_{\min}(u,v) \geq 0$ with equality if and only if $u = v$
- $d^M_{\min}(u,w) + d^M_{\min}(v,w) \geq d^M_{\min}(u,v)$ with equality if and only if w is on the (possibly non-unique) shortest path between u and v
- if u and v are disconnected in $\mathcal{M}^+$ then, by convention, $d^M_{\min}(u,v) = \infty$

Theorem 7.3: Let r, s and t be nodes in some block B (of size ≥ 3) in the triangulated graph $\mathcal{M}^+$ with $d^M_{\min}(r,s) \leq d^M_{\min}(r,t)$. Then for any v on an induced path between s and t we have

$$d^M_{\min}(r,v) \leq d^M_{\min}(r,t)$$

Proof: Since the nodes are all in the same block we know that there must be at least two paths between any two nodes. Let p and q be the shortest paths from nodes r to s and r to t, respectively (denoted $r \xrightarrow{p} s$ and $r \xrightarrow{q} t$). If p and q meet at some node other than r then they will share the path from that node to r (otherwise they cannot be shortest paths). Let such a shared node furthest from r be r'. Then $d^M_{\min}(r,t) = d^M_{\min}(r,r') + d^M_{\min}(r',t)$ and $d^M_{\min}(r,v) \leq d^M_{\min}(r,r') + d^M_{\min}(r',v)$ so if the result holds for r' it holds for r. Without loss of generality assume that there is no such r'. Now consider the following cases:

- If q contains v then $d^M_{\min}(r,v) = d^M_{\min}(r,t) - d^M_{\min}(v,t) < d^M_{\min}(r,t)$.
- If p contains v then $d^M_{\min}(r,v) = d^M_{\min}(r,s) - d^M_{\min}(v,s) < d^M_{\min}(r,s) \leq d^M_{\min}(r,t)$.
- Otherwise v is on some other (induced) path between s and t. But now $r \xrightarrow{p} s \xrightarrow{} v \xrightarrow{} t \xrightarrow{q} r$ forms a cycle of length ≥ 4. Since $\mathcal{M}^+$ is triangulated there must be an edge from v to some node on p or q. There cannot be an edge between s and t or else there would not be any induced paths between s and t. But then $d^M_{\min}(r,v) \leq d^M_{\min}(r,t)$.

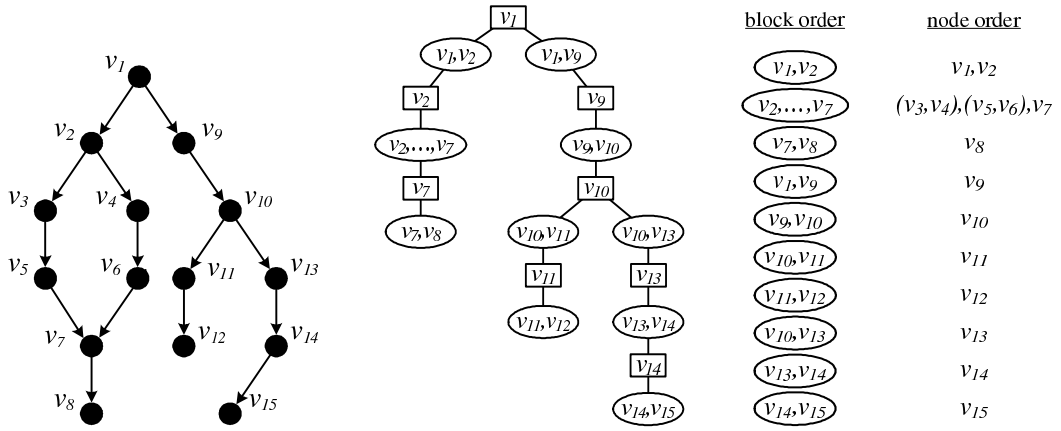


∎

Algorithm 7: Block-Shortest-Path Ordering

Input : $\mathcal{G}$ // input Bayesian network
$\mathcal{M}^+$ // corresponding moralized triangulation
Output: O // an ordering $X_1,\ldots,X_N$

$O \leftarrow \emptyset$
$O_T \leftarrow$ topological ordering of the nodes in $\mathcal{G}$
$O_B \leftarrow$ depth-first search ordering of blocks in $\mathcal{M}^+$
while $O_B \neq \emptyset$ **do**
  $B \leftarrow$ pop O_B
  $R \leftarrow$ cut-vertex of B with lowest O_T
  Push nodes in B to O in order of $(O_T, d^M_{\min}(R,\cdot))$
end
return O



(a) Bayesian Network, $\mathcal{G}$ (b) Block Tree, $\mathcal{B}$ (c) BSP Ordering

Figure 7: Concrete example of BSP ordering using the Bayesian network from Figure 4. Nodes in parentheses are the same distance from the root cut-vertex and can be ordered arbitrarily.

We now use this result to order nodes according to their distance from the cut-vertex in the block that connects it to the blocks already ordered (which we call the root cut-vertex). Algorithm (7) shows how our Block-Shortest-Path (BSP) ordering is constructed and Figure 7 demonstrates the application of that ordering to a concrete example.

Finally, we note that the above ordering, while almost strict, still allows for variables that are the same distance from the root cut-vertex of the block to be ordered arbitrarily. Indeed, as the following example shows two nodes that are the same distance from the block cut-vertex can be symmetrically contaminating. We break such ties arbitrarily.

Example 7.4: Consider, again, the example network shown in Example 7.2. The set of nodes $\{v_2, v_3, v_6, v_7, v_8\}$ are all the adjacent to r and so can be ordered arbitrarily. An edge from v_2 to v_8 (or vice versa) will contaminate v_7. Likewise an edge from v_3 to v_6 (or vice versa) will also contaminate v_7. It turns out that for any ordering of these nodes, it is always possible to add an edge that will contaminate other nodes in the set. This is consistent with the contamination result of Theorem 7.3 since these nodes are all equi-distant from r. ∎

8. Experimental Evaluation

In this section we perform experimental validation of our approach and show that it is beneficial for learning Bayesian networks of bounded treewidth. Specifically, we demonstrate that by making use of global structure modification steps, our approach leads to superior generalization. In order to evaluate our method we compare against two strong baseline approaches.

The first baseline is an improved variant of the thin junction tree approach of Bach and Jordan (2002). We start, as in our method, with a Chow-Liu forest and iteratively add the single best scoring edge. To make the approach as comparable to ours as possible, at each iteration, we triangulate the model using either the maximum cardinality search or minimum fill-in heuristics (see, for example, Koster et al., 2001), as well as using our treewidth friendly triangulation, and take the triangulation that results in a lower treewidth.[5] As in our method, when the treewidth bound is reached, we continue to add edges that improve the model selection score until no such edges can be found that do not also increase the treewidth bound.

The second baseline is an aggressive structure learning approach that combines greedy edge modifications with a TABU list (e.g., Glover and Laguna, 1993) and random moves. This approach is not constrained by a treewidth bound. Comparison to this baseline allows us to evaluate the merit of our method with respect to an unconstrained state-of-the-art search procedure.

We evaluate our method on four real-world data sets that are described below. Where relevant we also compare our results to the results of Chechetka and Guestrin (2008).

8.1 Gene Expression

In our first experiment, we consider a continuous data set based on a study that measures the expression of the baker's yeast genes in 173 experiments (Gasch et al., 2000). In this study, researchers measured the expression of 6152 yeast genes in response to changes in the environmental conditions, resulting in a matrix of 173×6152 measurements. The measurements are real-valued and, in our experiments, we learn sigmoid Bayesian networks using the Bayesian Information Criterion (BIC) (Schwarz, 1978) for model selection. For practical reasons, we consider the fully observed set of 89 genes that participate in general metabolic processes (Met). This is the larger of the two sets used by Elidan et al. (2007), and was chosen since part of the response of the yeast to changes in its environment is in altering the activity levels of different parts of its metabolism. We treat the genes as variables and the experiments as instances so that the learned networks indicate possible regulatory or functional connections between genes (Friedman et al., 2000).

Figure 8 shows test log-loss results for the 89 variable gene expression data set as a function of the treewidth bound. The first obvious phenomenon is that both our method (solid blue squares) and

5. We note that in all of our experiments there was only a small difference between the minimum fill-in and maximum cardinality search heuristics for upper bounding the treewidth of the model at hand.

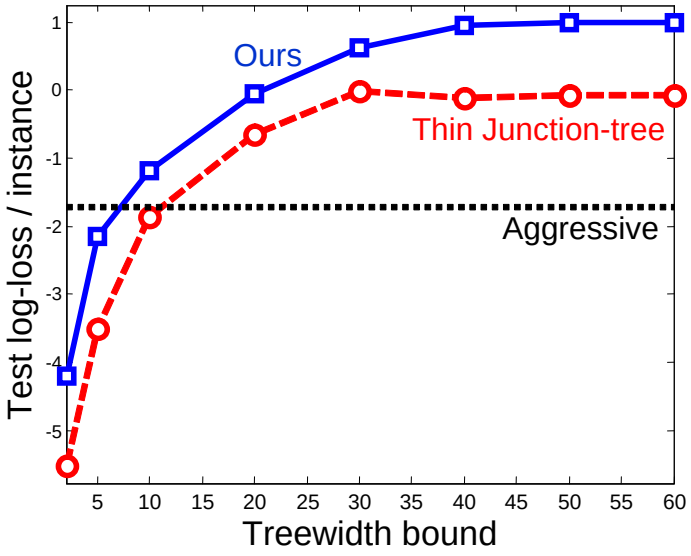


Figure 8: Average test set log-loss per instance over five folds (y-axis) versus the treewidth bound (x-axis) for the 89 variable gene expression data set. Compared are our method (solid blue squares) with the **Thin junction tree** approach (dashed red circles), and an **Aggressive** greedy approach of unbounded treewidth that also uses a TABU list and random moves (dotted black).

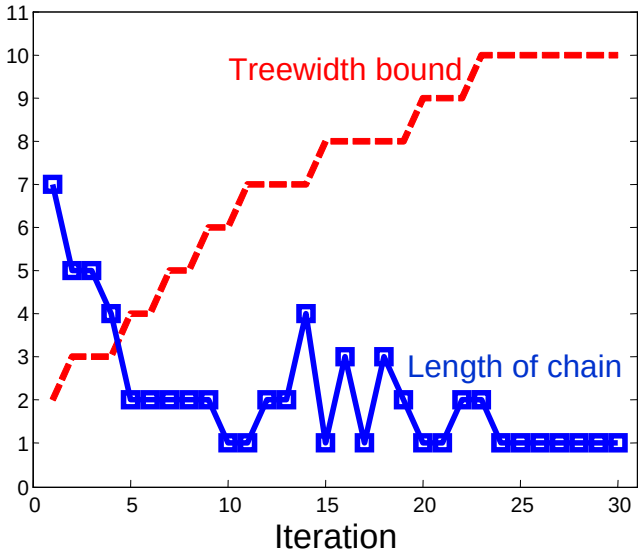


Figure 9: Plot showing the number of edges (in the learned chain) added during each iteration for a typical run with treewidth bound of 10 for the 89 variables gene expression data set. The graph also shows our treewidth estimate at the end of each iteration.

the greedy junction tree approach (dashed red circles) are superior to the aggressive baseline (dotted black). As one might expect, the aggressive baseline achieves a higher BIC score on training data (not shown), but overfits due to the scarcity of the data. By greedy edge addition (the junction tree approach) or global chain addition (our approach), this overfitting is avoided. Indeed, a better choice of edges, that is, ones chosen using a global operator, can lead to increased robustness and better generalization. This is evident by the consistent superiority of our method (solid blue squares) over the greedy variant (dashed red circles). Importantly, even when the treewidth bound is increased passed the saturation point our method surpasses both the thin junction tree approach of Bach and Jordan (2002) and the aggressive search strategy. In this case, we are learning unbounded Bayesian networks and all of the benefit comes from the global nature of our structure modifications.

To qualitatively illustrate the progression of our algorithm from iteration to iteration, we plot the number of edges in the chain (solid blue squares) and treewidth estimate (dashed red) at the end of each iteration. Figure 9 shows a typical run for the 89 variable gene expression data set with treewidth bound set to 10. Our algorithm aggressively adds many edges (making up an optimal chain) per iteration until parts of the network reach the treewidth bound. At that point (iteration 24) the algorithm resorts to adding the single best edge per iteration until no more edges can be added without increasing the treewidth (or that have an adverse effect on the score). To appreciate the non-triviality of some of the chains learned with 4, 5 or 7 edges, we recall that the example shows edges added *after* a Chow-Liu model was initially learned. It is also worth noting that despite their complexity, some chains do not increase the treewidth estimate and for a given treewidth bound K, we typically have more than K iterations (in this example 24 chains are added before reaching the treewidth bound). The number of such iterations is still polynomially bounded as for a Bayesian network with N variables adding more than $K \cdot N$ edges will necessarily result in a treewidth that is greater than K.

In order to verify the efficiency of our method we measured the running time of our algorithm as a function of treewidth bound. Figure 10 shows results for the 89 variable gene expression data set. Observe that our method (solid blue squares) and the greedy thin junction tree approach (dashed red circles) are both approximately linear in the treewidth bound. Appealingly, the additional computation required by our method is not significant and the differences between the two approaches are at most 25%. This should not come as a surprise since the bulk of the time is spent on the collection of sufficient statistics from the data.

It is also worthwhile to discuss the range of treewidths considered in the above experiment as well as the Haplotype sequence experiment considered below. While treewidths of 30 and beyond may seem excessive for exact inference, state-of-the-art exact inference techniques (e.g., Darwiche, 2001; Marinescu and Dechter, 2005) can often handle inference in such networks (for some examples see Bilmes and Dechter, 2006). Since, as shown in Figure 8, it is beneficial to learn models with large treewidth, methods such as ours for learning and the state-of-the-art techniques for inference allow practitioners to push the envelope of the complexity of models learned for real applications.

8.2 The Traffic and Temperature Data Sets

We now compare our method to the mutual-information based LPACJT method for learning bounded treewidth model of Chechetka and Guestrin (2008) (we compare to better of the variants presented in that work). While providing theoretical guarantees (under some assumptions), their method is exponential in the treewidth and cannot be used in a setting similar to the gene expression experi-

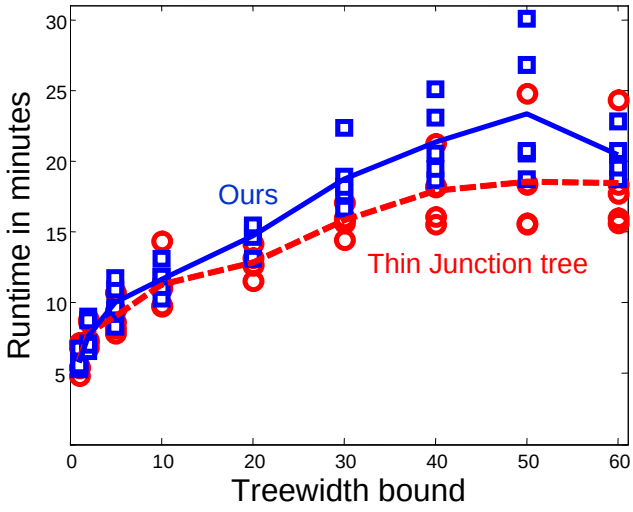
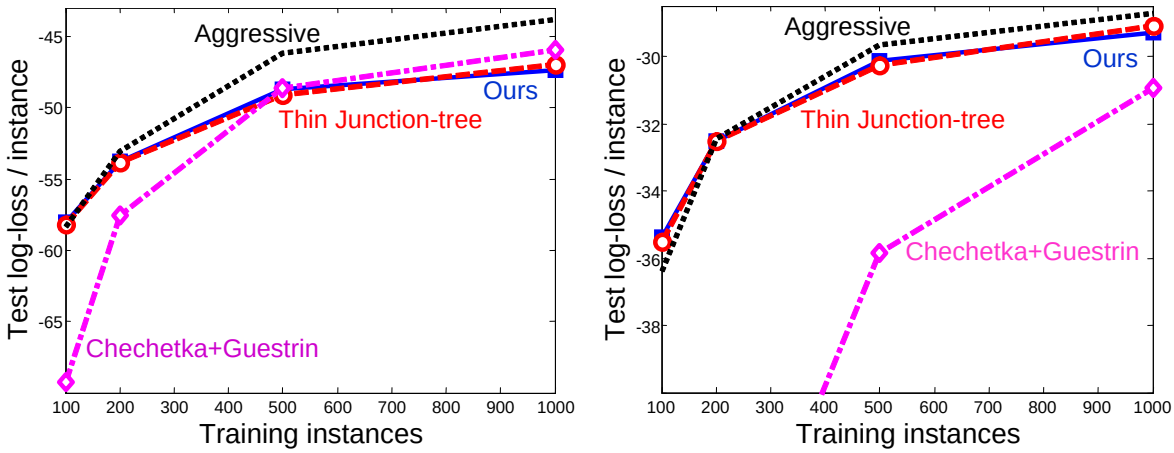


Figure 10: Running time in minutes on the 89 variable gene expression data set (y-axis) as a function of treewidth bound (x-axis). The graph compares our method (solid blue squares) with the thin junction tree approach (dashed red circles). The markers show times for the 5 different fold runs for each treewidth while the line shows the average running time.

ment above. Instead, we compare on the two discrete real-life data set considered in Chechetka and Guestrin (2008). The temperature data is from a two-month deployment of 54 sensor nodes (15K data points) (Deshpande et al., 2004) where each variable was discretized into 4 bins. The traffic data set contains traffic flow information measured every five minutes in 32 locations in California for one month (Krause and Guestrin, 2005). Values were discretized into 4 bins. For both data sets, to make the comparison fair, we used the same discretization and train/test splits as in Chechetka and Guestrin (2008). Furthermore, as their method can only be applied to a small treewidth bound, we also limited our model to a treewidth of two. Figure 11 compares the different methods. Both our method and the thin junction tree approach significantly outperform the LPACJT on small sample size. This result is consistent with that reported in Chechetka and Guestrin (2008) and is due to the fact that the LPACJT method does not naturally use regularization which is crucial in the sparse-data regime. The performance of our method is comparable to the greedy thin junction tree approach with no obvious superiority to either method. This should not come as a surprise since the fact that the unbounded aggressive approach is not significantly better suggests that the strong signal in the data can be captured rather easily. In fact, Chechetka and Guestrin (2008) show that even a Chow-Liu tree does rather well on these data sets (compare this to the gene expression data set where the aggressive variant was superior even at a treewidth of four).

8.3 Haplotype Sequences

Finally we consider a more difficult discrete data set consisting of a sequence of binary single nucleotide polymorphism (SNP) alleles from the Human HapMap project (Consortium, 2003). Our



(a) Temperature (54 variables)

(b) Traffic (32 variables)

Figure 11: Average test set log-loss per instance over five folds (y-axis) versus the number of training instances (x-axis) for the temperature and traffic data sets. Compared are our method (solid blue squares), the **Thin junction tree** approach (dashed red circles), an **Aggressive** greedy approach of unbounded treewidth that also uses a TABU list and random moves (dotted black), and the mutual-information based method of Chechetka and Guestrin (2008) (dash-dot magenta diamonds). For all of our methods except the unbounded **Aggressive**, the treewidth bound was set to two.

model is defined over 200 SNPs (variables) from chromosome 22 of a European population consisting of 60 individuals.[6] In this case, there is a natural ordering of variables that corresponds to the position of the SNPs in the DNA sequence. Figure 12 shows test log-loss results when this ordering is enforced (thicker lines) and when it is not (thinner) lines. Our small benefit over the greedy thin junction tree approach of Bach and Jordan (2002) when the treewidth bound is non-trivial (>2) grows significantly when we take advantage of the natural variable order. Interestingly, this same order decreases the performance of the thin junction tree method. This should not come as a surprise as the greedy method does not make use of a node ordering, while our method provides optimality guarantees with respect to a variable ordering at each iteration. Whether constrained to the natural variable ordering or not, our method ultimately also surpasses the performance of the aggressive unbounded search approach.

9. Discussion and Future Work

In this work we presented a novel method for learning Bayesian networks of bounded treewidth in time that is polynomial in *both* the number of variables and the treewidth bound. Our method builds on an edge update algorithm that dynamically maintains a valid moralized triangulation in a way that facilitates the addition of chains that are guaranteed to increase the treewidth by at most one. We demonstrated the effectiveness of our treewidth-friendly method on real-life data sets, and

6. We considered several different sequences along the chromosome with similar results.

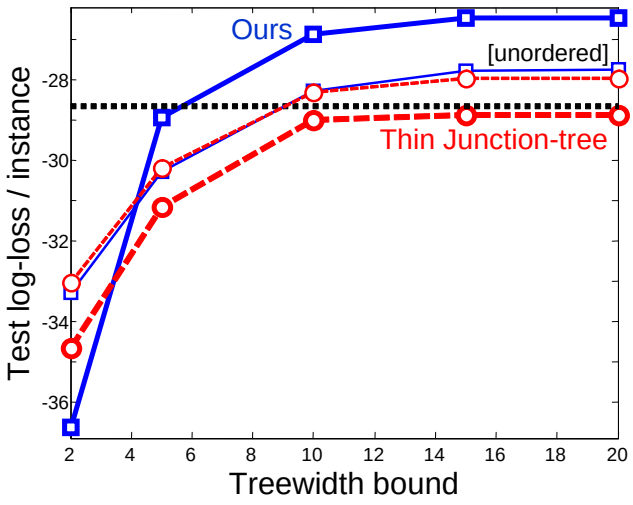


Figure 12: Average test set log-loss per data instance over five folds (y-axis) versus the treewidth bound (x-axis) for the 200 variable Hapmap data set. The graph compares our method (solid blue squares) with the greedy approach (dashed red circles), and an aggressive greedy approach of unbounded treewidth that also uses a TABU list and random moves (dotted black). The thicker lines show the results for a fixed ordering of the variables according to the location along the DNA sequence. The thinner lines show the results without any constraint on the node ordering.

showed that by using global structure modification operators, we are able to learn better models than competing methods even when the treewidth of the models learned is not constrained.

Our method can be viewed as a generalization of the work of Chow and Liu (1968) that is constrained to a chain structure but that provides an optimality guarantee (with respect to a node ordering) at every treewidth. In addition, unlike the thin junction trees approach of Bach and Jordan (2002), we also provide a guarantee that our estimate of the treewidth bound will not increase by more than one at each iteration. Furthermore, we add multiple edges at each iteration, which in turn allows us to better cope with the problem of local maxima in the search. To our knowledge, ours is the first method for efficiently learning bounded treewidth Bayesian networks with structure modifications that are not fully greedy.

Several other methods aim to generalize the work of Chow and Liu (1968). Karger and Srebro (2001) propose a method that is guaranteed to learn a good approximation of the optimal Markov network given a treewidth bound. Their method builds on a hyper-graph that is exponential in the treewidth bound. Chechetka and Guestrin (2008) also propose an innovative method with theoretical guarantees on the quality of the learned model (given some mild assumptions on the generating distribution), but in the context of Bayesian networks. However, like the approach of Karger and Srebro (2001), the method is exponential in the treewidth bound. Thus, both approaches are only practical for treewidths that are much smaller than the ones we consider in this work. In addition, the work of Chechetka and Guestrin (2008) does not naturally allow for the use of regularization.

This has significant impact on performance when the number of training samples is limited, as demonstrated in Section 8.

Meila and Jordan (2000) suggested the use of a mixture of trees, generalizing the Chow-Liu tree on an axis that is orthogonal to a more complex Bayesian network. They provide an efficient method for obtaining a (penalized) likelihood local maxima but their work is limited to a particular and relatively simple structure. Dasgupta (1999) suggested the use of poly-trees but proved that learning the optimal poly-tree is computationally difficult. Other works study this question but in the context where the *true distribution* is assumed to have bounded treewidth (e.g., Beygelzimer and Rish, 2004; Abbeel et al., 2006, and references within).

Our method motivates several exciting future directions. It would be interesting to see to what extent we could overcome the limitation of having to commit to a specific node ordering at each iteration. While we provably cannot consider any node ordering, it may be possible to polynomially provide a reasonable approximation. Second, it may be possible to refine our characterization of the contamination that results from an edge update, which in turn may facilitate the addition of more complex treewidth-friendly structures at each iteration. Finally, we are most interested in exploring whether tools similar to the ones employed in this work could be used to dynamically update the bounded treewidth structure that is the approximating distribution in a variational approximate inference setting.

Acknowledgments

We are grateful to Ben Packer for many useful discussions and comments. Much of the current work was carried out while Gal Elidan was in Stanford University.

References

P. Abbeel, D. Koller, and A. Y. Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7:1743–1788, 2006.

F. Bach and M. I. Jordan. Thin junction trees. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, Mass., 2002. MIT Press.

A. Beygelzimer and I. Rish. Approximability of probability distributions. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

J. Bilmes and R. Dechter. Evaluation of probabilistic inference, the twenty second conference on uncertainty in artificial intelligence. `ssli.ee.washington.edu/∼bilmes/UAI06InferenceEvaluation`, 2006.

H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems 20*, pages 273–280. MIT Press, Cambridge, MA, 2008.

D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, New York, 1996.

C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory*, 14:462–467, 1968.

The International HapMap Consortium. The international hapmap project. *Nature*, 426:789–796, 2003.

G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

P. Dagum and M. Luby. An optimal approximation algorithm for baysian inference. *Artificial Intelligence*, 60:141–153, 1993.

A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 126, 2001.

S. Dasgupta. Learning polytrees. In K. Laskey and H. Prade, editors, *Proc. Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI '99)*, pages 134–141, San Francisco, 1999. Morgan Kaufmann.

A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Very Large Data Bases (VLDB) Conference*, 2004.

R. Diestel. *Graph Theory*. Springer, 3rd edition, 2005.

G. A. Dirac. On rigid circuit graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 25, Universität Hamburg, 1961.

G. Elidan, I. Nachman, and N. Friedman. "ideal parent" structure learning for continuous variable bayesian networks. *Journal of Machine Learning Research*, 8:1799–1833, 2007.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

A. Gasch, P. Spellman, C. Kao, O. Carmel-Harel, M. Eisen, G. Storz, D. Botstein, and P. Brown. Genomic expression program in the response of yeast cells to environmental changes. *Molecular Biology of the Cell*, 11:4241–4257, 2000.

F. Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.

D. Heckerman. A tutorial on learning with Bayesian networks. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands, 1998.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

D. Karger and N. Srebro. Learning markov networks: maximum bounded tree-width graphs. In *Symposium on Discrete Algorithms*, pages 392–401, 2001.

A. Koster, H. Bodlaender, and S. Van Hoesel. Treewidth: Computational experiments. Technical report, Universiteit Utrecht, 2001.

A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In F. Bacchus and T. Jaakkola, editors, *Proc. Twenty First Conference on Uncertainty in Artificial Intelligence (UAI '05)*, San Francisco, 2005. Morgan Kaufmann.

W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. of the Royal Statistical Society*, B 50(2):157–224, 1988.

R. Marinescu and R. Dechter. And/or branch-and-bound for graphical models. *IJCAI*, 2005.

C. Meek. Finding a path is harder than finding a tree. *Journal of Artificial Intelligence Research*, 15:383–389, 2001.

M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.

M. Narasimhan and J. Bilmes. Pac-learning bounded tree-width graphical models. In M. Chickering and J. Halpern, editors, *Proc. Twenieth Conference on Uncertainty in Artificial Intelligence (UAI '04)*, San Francisco, 2003. Morgan Kaufmann.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

N. Robertson and P. D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1987.

G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

R. Tarjan and M. Yannakakis. Simple linear time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13:3:566–579, 1984.

Journal of Machine Learning Research 9 (2008) 2733-2759 Submitted 10/07; Revised 7/08; Published 12/08

Automatic PCA Dimension Selection for High Dimensional Data and Small Sample Sizes

David C. Hoyle DAVID.HOYLE@MANCHESTER.AC.UK
North West Institute for BioHealth Informatics,
University of Manchester, Faculty of Medical and Human Sciences,
University Place (East), Oxford Rd., Manchester, M13 9PL, UK.

Editor: Chris Williams

Abstract

Bayesian inference from high-dimensional data involves the integration over a large number of model parameters. Accurate evaluation of such high-dimensional integrals raises a unique set of issues. These issues are illustrated using the exemplar of model selection for principal component analysis (PCA). A Bayesian model selection criterion, based on a Laplace approximation to the model evidence for determining the number of signal principal components present in a data set, has previously been show to perform well on various test data sets. Using simulated data we show that for d-dimensional data and small sample sizes, N, the accuracy of this model selection method is strongly affected by increasing values of d. By taking proper account of the contribution to the evidence from the large number of model parameters we show that model selection accuracy is substantially improved. The accuracy of the improved model evidence is studied in the asymptotic limit $d \to \infty$ at fixed ratio $\alpha = N/d$, with $\alpha < 1$. In this limit, model selection based upon the improved model evidence agrees with a frequentist hypothesis testing approach.

Keywords: PCA, Bayesian model selection, random matrix theory, high dimensional inference

1. Introduction

The generation of high dimensional data is fast becoming a common place occurrence. Examples range from genomics and molecular biology, for example high-throughput single nucleotide polymorphism (SNP) genotyping scans (Price et al., 2006) and microarray gene expression studies (Golub et al., 1999), to geophysical imaging, for example hyperspectral image data (Landgrebe, 2002). Intuitive visualization of the data and construction of novel features from the data are key tasks in processing such high-dimensional data. This often involves dimensionality reduction, for which a number of algorithms exist. Principal component analysis (PCA) is a ubiquitous method of data analysis and dimensionality reduction (Joliffe, 1986). Its utility and success stems from the simplicity of the method - one simply calculates the eigenvectors and eigenvalues of the sample covariance matrix $\hat{C}$ of the data set. A subset of the eigenvectors of $\hat{C}$, the principal components, are then selected to represent the data. A 'kernelized' version has been formulated - kernel PCA (Scholköpf, Smola, and Müller, 1998), and building on probabilistic formulations (Roweis, 1998; Tipping and Bishop, 1999a) it has also been extended to a mixture of principal component analysers (Tipping and Bishop, 1999b). In the latter case a number of local linear models are embedded in the high dimensional data space, with the properties of each local model being determined from the local responsibility-weighted covariance matrix.

Clearly selection of the correct number of principal components is crucial to the success of PCA in representing a data set. Identification of the appropriate signal dimensionality is just a model selection process to which the techniques of Bayesian model selection can be applied via a suitable approximation of the Bayesian evidence (MacKay, 1992). What is the most suitable method of approximating the evidence for high-dimensional data and what are the inherent problems? These are the research questions we address and a roadmap for the paper is given below,

• In Section 2 we motivate why high-dimensional small sample size data sets present a challenge for Bayesian model selection.

• In Section 3 we summarize the behaviour of the eigenvectors and eigenvalues of sample covariance matrices formed from high-dimensional small sample size data sets.

• In Section 4.1 we review the formalism of Bayesian model selection for PCA, and evaluate through simulation the model selection accuracy of an existing approximation to the Bayesian evidence.

• In Section 4.2 we develop an improved approximation to the Bayesian evidence specifically for high dimensional data.

• In Section 5 we evaluate the asymptotic properties of the improved approximation to the model evidence.

• In Section 6 the model selection performance of the improved approximation to the model evidence is compared with a frequentist hypothesis testing approach to model selection.

2. The Challenge of High-Dimensional Data for Bayesian Model Selection

A number of Bayesian formulations of PCA have followed from the probabilistic formulation of Tipping and Bishop (1999a), with the necessary marginalization being approximated through both Laplace approximations (Bishop, 1999a; Minka, 2000, 2001a) and variational bounds (Bishop, 1999b). More recently, work within the statistics research community has used a Bayesian variational approach to derive an explicit conditional probability distribution for the signal dimension given the data (Šmídl and Quinn, 2007). However, these results have only been tested on low dimensional data with relatively large sample sizes. A somewhat more tractable expression for the signal dimension posterior was also obtained by Minka (2000, 2001a) and it is that Bayesian formulation of PCA that we draw upon. By performing a Laplace approximation (Wong, 1989), that is, expanding about the maximum posterior solution, Minka derived an elegant approximation to the probability, the model evidence $p(D|k)$, of observing a data set D given the number of principal components k (Minka, 2000, 2001a). The signal dimensionality of the given data set is then estimated by the value of k that maximizes $p(D|k)$. As with any Bayesian model selection procedure, if the data has truly been generated by a model of the form proposed, then one is guaranteed to select the correct model dimensionality as the sample increases to an infinite size. Minka's dimensionality selection method performs well when tested on data sets of moderate size and dimensionality. Indeed, the Laplace approximation incorporates the leading order term in an asymptotic expansion of the Bayesian evidence, with the sample size N playing the role of the 'large' parameter, and so we would expect the Laplace approximation to be increasingly accurate as $N \to \infty$. In real-world data sets, such as those emanating from molecular biology experiments, the number of variables d is often very much greater than the sample size N, with $d \sim 10^4$ yet $N \sim 10$ or $N \sim 10^2$ not uncommon (Hoyle and Rattray, 2003). Typically, data sets with a sample size of $N = 100$ might be considered as large enough to be well approximated by the asymptotic limit $N \to \infty$, and therefore the Laplace

approximation to be appropriate. However, though retaining only a small number of terms from the asymptotic expansion of the evidence would be increasingly accurate as $N \to \infty$, individual expansion coefficients may be significant due to the large data dimensionality d. This suggests that for real finite sample size data sets, higher order terms in the asymptotic expansion not encapsulated within the Laplace approximation will make significant contributions to the evidence, and model selection based upon a simplistic application of the Laplace approximation will perform poorly. What then defines a 'large' sample size N is clearly dependent on the data dimensionality d. We would expect the conjectures about the previously derived Laplace approximation to the evidence to be increasingly true when the data dimensionality is very much larger than the sample size, that is, $N \ll d$, the situation encountered for many modern data sets. For high dimensional data, rather than considering the evidence to be close to its value obtained in the asymptotic limit $N \to \infty$ at fixed d, it may be more appropriate to consider the evidence as being close to its value in the distinguished limit $d, N \to \infty$ at fixed $\alpha = N/d$. Within this paradigm, developing a suitable Gaussian approximation requires us to identify all contributions to the evidence that would scale extensively, that is increase linearly with N, as $N, d \to \infty$ at fixed α. This would be increasingly important for $\alpha < 1$, where the contribution to the evidence resulting from many features can be significant. Ideally we should re-formulate the evidence as an integration over a set of variables which remains finite in number in the distinguished limit.

To be more explicit, consider that the Bayesian approach to model selection in PCA starts from the probability $p(D|k,\boldsymbol{\theta})p(\boldsymbol{\theta}|k)$ and integrates over the model parameters $\boldsymbol{\theta}$ to obtain the evidence $p(D|k)$. This integration is often evaluated by the aforementioned Laplace approximation - expansion about the maximum of $p(D|k,\boldsymbol{\theta})p(\boldsymbol{\theta}|k)$ and evaluation of the consequent tractable Gaussian integrals. For high-dimensional data the model parameters may consist of a small set of parameters, $\boldsymbol{\theta}_k$, of order of the signal dimensionality k, and a much larger set of parameters, $\boldsymbol{\theta}_d$, of order of the data dimensionality. For example, the latter may be the principal vectors, in the d-dimensional space, that form part of the model. Overall we can write $\boldsymbol{\theta} = (\boldsymbol{\theta}_d, \boldsymbol{\theta}_k)$. Integration over $\boldsymbol{\theta}_d$ provides a significant contribution to $p(D|k)$ due simply to the large number of individual model parameters that we are integrating over. In this scenario, the values of $\boldsymbol{\theta}_k$ obtained from maximizing $\int p(D|k,\boldsymbol{\theta}_d,\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_d,\boldsymbol{\theta}_k|k)d\boldsymbol{\theta}_d$ and $p(D|k,\boldsymbol{\theta}_d,\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_d,\boldsymbol{\theta}_k|k)$ do not coincide. In fact for large values of d they may be significantly different. The more accurate estimates of $\boldsymbol{\theta}_k$ are naturally obtained from the maximum of $\int p(D|k,\boldsymbol{\theta}_d,\boldsymbol{\theta}_k)p(\boldsymbol{\theta}_d,\boldsymbol{\theta}_k|k)d\boldsymbol{\theta}_d$, and consequently the more accurate estimates of the evidence $p(D|k)$ are obtained by expanding about this maximum.

The distorting effects of high dimensionality upon covariance matrix eigenvalue spectra and eigenvectors are well known from random matrix theory (RMT) studies (Johnstone, 2006). The RMT studies inform us about the expected sample covariance eigenvalue spectrum in the limit $d \to \infty$ (at fixed α), and consequently the limits of any model selection procedure based upon the observed eigenvalue spectra. As PCA is based upon the eigenvalues and eigenvectors of $\hat{\boldsymbol{C}}$, understanding their behaviour for small sample sizes and high data dimensions is key to understanding the behaviour of the existing model selection criterion, including the Bayesian model selection approach of Minka. Results from RMT studies are summarized in Section 3.

3. High-Dimensional Sample Covariance Matrices

We envisage a scenario where one has N, d-dimensional data vectors $\boldsymbol{\xi}_\mu, \mu = 1, \ldots, N$, with sample mean $\bar{\boldsymbol{\xi}}$, which are drawn from a multi-variate Gaussian distribution with covariance $\boldsymbol{C}$. The eigen-

values of $\boldsymbol{C}$ we denote by Λ_i, $i = 1,\ldots,d$. The sample data vectors $\boldsymbol{\xi}_\mu$ contain both signal and noise components so we represent,

$$\boldsymbol{C} = \sigma^2 \boldsymbol{I} + \sum_{m=1}^{S} \sigma^2 A_m \boldsymbol{B}_m \boldsymbol{B}_m^T \quad , \quad \boldsymbol{B}_m^T \boldsymbol{B}_{m'} = \delta_{mm'} \quad , \quad A_m \geq 0 \; \forall m \; , \tag{1}$$

corresponding to a population covariance $\boldsymbol{C}$ that contains a small number, S, of orthogonal signal components, $\{\boldsymbol{B}_m\}_{m=1}^S$, but that is otherwise isotropic. Here, σ^2 represents the variance of the additive noise component of the sample data vectors. Such models have been termed "spiked" covariance models within the statistics research literature (Johnstone, 2001), due to the small number of δ-function spikes in the population covariance eigenspectrum. In this case the population eigenvalues are $\Lambda_i = \sigma^2(1+A_i)$, $i \leq S$ and $\Lambda_i = \sigma^2$, $i > S$. The signal strengths $\sigma^2 A_m$ merely determine the population covariance eigenvalues corresponding to signal directions, and so the number of signal components S is commonly estimated by some process of inspection of the ordered eigenvalues $\lambda_i, i = 1,\ldots,d$, of the sample covariance matrix $\hat{\boldsymbol{C}} = N^{-1}\sum_\mu (\boldsymbol{\xi}_\mu - \bar{\boldsymbol{\xi}})(\boldsymbol{\xi}_\mu - \bar{\boldsymbol{\xi}})^T$.

When the sample size is greater than the dimensionality, that is, $N > d$, the sample covariance eigenvalues λ_i may be reasonable estimators of the population covariance eigenvalues Λ_i, and indeed are asymptotically unbiased estimators, that is, $\lambda_i \to \Lambda_i$ as $N \to \infty$ for fixed dimensionality d (Anderson, 1963). However, for small sample sizes $N \leq d$ the sample covariance $\hat{\boldsymbol{C}}$ is singular with a $d - N + 1$ degenerate zero eigenvalue. Similarly, the non-zero sample covariance eigenvalues, λ_i, $i = 1,\ldots,N-1$, can display considerable bias. This is reflected in the expected eigenspectrum, $\rho(\lambda)$, which is simply defined as the expectation over data sets of the empirical eigenvalue density,

$$\rho(\lambda) \; = \; \mathrm{E}_{\boldsymbol{\xi}} \left(\frac{1}{d} \sum_{i=1}^{d} \delta(\lambda - \lambda_i) \right) \; .$$

Here $\delta(x)$ is the Dirac δ-function, and we have used $\mathrm{E}_{\boldsymbol{\xi}}(\cdot)$ to denote expectation over the ensemble of sample data sets. The empirical eigenvalue density is considered to be a self-averaging quantity, such that as $N \to \infty$ the eigenvalue density from any individual sample covariance matrix is well represented by the ensemble average. Therefore, for large sample covariance matrices studying the behaviour of the expected sample covariance eigenvalue distribution provides us with insight into the behaviour of individual sample covariance matrices and consequently the behaviour of any model selection algorithms based upon the sample covariance eigenvalues.

When no signal components are present, that is, $\boldsymbol{C} = \sigma^2 \boldsymbol{I}$, and in the limit $d \to \infty$ with $\alpha = N/d$ fixed, the expected distribution of sample eigenvalues tends to the Marčenko-Pastur distribution (Marčenko and Pastur, 1967),

$$\begin{aligned} \rho(\lambda) \; = \; \rho_{bulk}(\lambda) \quad &= \quad (1-\alpha)\Theta(1-\alpha)\delta(\lambda) \\ &+ \quad \frac{\alpha}{2\pi\lambda\sigma^2} \sqrt{\max[0, (\lambda - \lambda_{min})(\lambda_{max} - \lambda)]} \; , \end{aligned} \tag{2}$$

where $\lambda_{max} = \sigma^2(1+\alpha^{-\frac{1}{2}})^2$, $\lambda_{min} = \sigma^2(1-\alpha^{-\frac{1}{2}})^2$, and $\Theta(x)$ is the Heaviside step function. Figure 1 shows examples of the Marčenko-Pastur distribution for different values of α. It should be noted that although the mean sample eigenvalue is an unbiased estimator of σ^2, that is, $\int_{0^-}^{\infty} d\lambda \, \lambda \rho_{bulk}(\lambda) = \sigma^2$, the individual non-zero sample covariance eigenvalues lie in the interval $[\lambda_{min}, \lambda_{max}]$ and so for $\alpha < 1$ are highly biased estimators of the corresponding population eigenvalues.

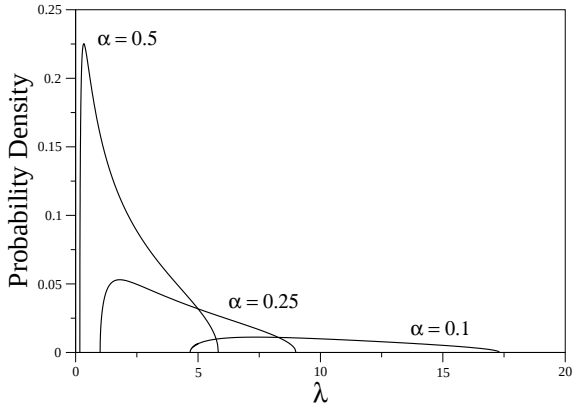


Figure 1: The Marčenko-Pastur limiting distribution for sample covariance eigenvalues, at $\alpha = 0.1, 0.25, 0.5$. In all cases $\sigma^2 = 1$. We have shown only the part of the distribution pertaining to non-zero eigenvalues. For $\alpha < 1$ there is also a δ-function peak at $\lambda = 0$ due to the singular nature of the sample covariance matrix - see main text.

Hoyle and Rattray (2004a) studied the expected behaviour of the sample covariance eigenvalue spectrum for "spiked" covariance models in the asymptotic limit $d \to \infty$ at fixed α, by using techniques from statistical physics. Similar results have been obtained within the statistics research community (Baik and Silverstein, 2006). As the addition of a small number, S, of signal directions provides a relatively small perturbation to an isotropic population covariance, the majority, or bulk of eigenvalues are still distributed according to the Marčenko-Pastur law. For this reason we have used $\rho_{bulk}(\lambda)$ to denote the Marčenko-Pastur distribution. For the "spiked" covariance models of Equation (1) the expected eigenvalue distribution $\rho(\lambda)$ is modified from $\rho_{bulk}(\lambda)$. At finite but large values of d and N the expected sample covariance eigenvalue density can be approximated by,

$$\begin{aligned}\rho(\lambda) &= (1-\alpha)\Theta(1-\alpha)\delta(\lambda) + \frac{1}{d}\sum_{m=1}^{S}\delta(\lambda-\lambda_u(A_m))\Theta(\alpha-A_m^{-2}) \\ &+ \left(1-d^{-1}\sum_{m=1}^{S}\Theta(\alpha-A_m^{-2})\right)\frac{\alpha}{2\pi\lambda\sigma^2}\sqrt{\max[0,(\lambda-\lambda_{min})(\lambda_{max}-\lambda)]}\ , \qquad (3)\end{aligned}$$

where $\lambda_u(A) = \sigma^2(1+A)(1+(\alpha A)^{-1})$. A number of interesting features are present in this spectrum. A transition occurs at $\alpha = A_m^{-2}$, such that for $\alpha > A_m^{-2}$ a sample eigenvalue located at $\lambda = \lambda_u(A_m)$ can be resolved separately from the remaining Marčenko-Pastur bulk of eigenvalues. Thus for S signal components within the "spiked" covariance model we can observe up to S transitions in the sample covariance eigenspectrum, on increasing α. The first transition point $\alpha = A_1^{-2}$ corresponds to the transition point in learning the leading signal direction $\boldsymbol{B}_1$. The scenario of learning a single signal component $\boldsymbol{B}_1$ of strength A_1 has been studied by Reimann et al. (1996), who

considered the behaviour (as $d \to \infty$ at fixed α) of the expectation value of R_1^2, where $R_1 = \boldsymbol{B}_1 \cdot \boldsymbol{J}_1$ is the overlap between the first principal component $\boldsymbol{J}_1$ of the sample covariance and $\boldsymbol{B}_1$. One observes the phenomenon of retarded learning whereby $R_1^2 = 0$ for $\alpha < A_1^{-2}$ and $R_1^2 > 0$ for $\alpha > A_1^{-2}$. This has been generalized to learning multiple orthogonal signals and one observes a separate retarded learning transition at $\alpha = A_m^{-2}$ for each of the overlaps $R_m^2 = (\boldsymbol{B}_m \cdot \boldsymbol{J}_m)^2$, where $\boldsymbol{J}_m$ is the m^{th} principal component (Hoyle and Rattray, 2007). That the ability to detect the signal components is reflected in the sample covariance eigenvalue structure (with retarded learning transitions coinciding with transitions in the eigenspectrum) demonstrates the utility of the sample covariance eigenspectrum for model selection. It also highlights that if the true signal dimensionality is S then asymptotically we have at most only S sample covariance eigenvalues separated from the Marčenko-Pastur bulk distribution, dependent on the value of α. If, for the given value of α, we have $\hat{S}$ eigenvalues separated from the Marčenko-Pastur bulk distribution, then the asymptotic equivalence of the observed sample covariance eigenspectra when $\boldsymbol{C}$ contains S signals or $\hat{S} \leq S$ signals means that no correct Bayesian model selection procedure can, asymptotically, select greater than $\hat{S}$ principal components (applying an Occam's Razor like argument), since both models are equally capable of explaining the observed eigenspectra. Equally, for sufficiently small α it is impossible, asymptotically, to distinguish the sample spectrum from one which has been generated from a model containing no signal structure, that is, from a population covariance $\boldsymbol{C} = \sigma^2 \boldsymbol{I}$. Within these constraints placed by the expected behaviour of the observed eigenspectra we now attempt to derive a suitable Bayesian model selection procedure that performs well in the distinguished asymptotic limit $N, d \to \infty$ at fixed α.

4. Bayesian Model Selection

In this section we summarize the Bayesian model selection procedure for PCA. We start in Section 4.1 by reproducing the formulation of the Bayesian model evidence as outlined by Minka (2000, 2001a) and the subsequent Laplace approximation. In Section 4.2 we re-express the evidence in a form that is more suitable for application of a Gaussian approximation when $d, N \to \infty$ at fixed $\alpha < 1$.

4.1 Laplace Approximation of Minka

The data vectors $\boldsymbol{\xi}_\mu$ are modelled as being drawn from a multi-variate Gaussian distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma} = v\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^T$. Thus $\boldsymbol{\Sigma}$ acts as a model of the true population covariance $\boldsymbol{C}$. The matrix $\boldsymbol{H}$ represents the signal considered present in the data and so is modelled as being due to a small number, k, of orthogonal signal components $\boldsymbol{u}_i$, $i = 1, \ldots, k$. Consequently we set,

$$\boldsymbol{H} = \boldsymbol{U}(\boldsymbol{L} - v\boldsymbol{I}_k)^{1/2}\boldsymbol{W} \ , \ \boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}_k \ , \ \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_k \ ,$$

where the columns of the orthonormal matrix $\boldsymbol{U}$ are formed from the vectors $\boldsymbol{u}_i$. The parameter v provides an estimator of the true population noise level σ^2. The diagonal matrix $\boldsymbol{L}$ has elements l_i, $i = 1, \ldots, k$, which represent estimators of the population covariance eigenvalues Λ_i. The orthonormal matrix $\boldsymbol{W}$ represents an irrelevant rotation within the subspace and is subsequently eliminated from the calculation. Model selection proceeds via the standard use of Bayes' theorem,

$$p(\boldsymbol{H}, \boldsymbol{m}, v|D) \ = \ \frac{p(D|\boldsymbol{H}, \boldsymbol{m}, v)p(\boldsymbol{H}, \boldsymbol{m}, v)}{p(D)} \ .$$

The signal dimensionality, k, is implicit in the matrix $\boldsymbol{H}$. With a non-informative prior, the mean $\boldsymbol{m}$ can be integrated out to yield the probability of observing the data set D given $\boldsymbol{H}$ and v (Minka, 2001a),

$$p(D|\boldsymbol{H},v) \;=\; N^{-d/2}(2\pi)^{-(N-1)d/2}|\boldsymbol{H}\boldsymbol{H}^T+v\boldsymbol{I}|^{-(N-1)/2}\exp\left(-\frac{N}{2}\mathrm{tr}((\boldsymbol{H}\boldsymbol{H}^T+v\boldsymbol{I})^{-1}\hat{\boldsymbol{C}})\right)\,.$$

Given a prior $p(\boldsymbol{U},\boldsymbol{W},\boldsymbol{L},v)$ the evidence for a signal dimensionality k is then,

$$p(D|k) \;=\; \int d\boldsymbol{U}d\boldsymbol{W}d\boldsymbol{L}dv\,p(D|\boldsymbol{U},\boldsymbol{W},\boldsymbol{L},v)p(\boldsymbol{U},\boldsymbol{W},\boldsymbol{L},v)\,.$$

The integration over the elements $l_i,\ i=1,\ldots,k$ is restricted to the region $l_i \geq 0 \forall i$. Similarly, the integration over $\boldsymbol{U}$ and $\boldsymbol{W}$ is over the entire space of $d\times k$ and $k\times k$ orthonormal matrices respectively. For the relevant integration over $\boldsymbol{U}$ this is equivalent to integration over the Stiefel manifold $V_k(\mathbb{R}^d)$ defined by the set of all orthonormal k-frames in $\mathbb{R}^d$ (James, 1954) .

Minka chooses a conjugate prior,

$$p(\boldsymbol{U},\boldsymbol{W},\boldsymbol{L},v) \;\propto\; |\boldsymbol{H}\boldsymbol{H}^T+v\boldsymbol{I}|^{-(\eta+2)/2}\exp(-\frac{\eta}{2}\mathrm{tr}((\boldsymbol{H}\boldsymbol{H}^T+v\boldsymbol{I})^{-1}))\,, \tag{4}$$

where the hyper-parameter η controls the sharpness of the prior. For a non-informative prior η should be small and ultimately we shall take $\eta\to 0^+$ in our resulting approximation to the evidence $p(D|k)$. With the prior given in Equation (4) the evidence is Minka (2000, 2001a),

$$\begin{aligned} p(D|k) \;&=\; \frac{\mathcal{N}_k(d)}{\mathrm{Area}(V_k(\mathbb{R}^d))}\int d\boldsymbol{U}d\boldsymbol{L}dv\,|\boldsymbol{H}\boldsymbol{H}^T+v\boldsymbol{I}|^{-(N+1+\eta)/2} \\ &\times\; \exp(-\frac{N}{2}\mathrm{tr}((\boldsymbol{H}\boldsymbol{H}^T+v\boldsymbol{I})^{-1}(\hat{\boldsymbol{C}}+N^{-1}\eta\boldsymbol{I})))\,, \end{aligned} \tag{5}$$

with,

$$\mathcal{N}_k(d) \;=\; \frac{N^{-d/2}(2\pi)^{-(N-1)d/2}}{\Gamma\left((\frac{1}{2}\eta+1)(d-k)-1\right)}\,(\eta(d-k)/2)^{(\frac{1}{2}\eta+1)(d-k)-1}\,\frac{1}{\Gamma(\eta/2)^k}\,(\eta/2)^{\eta k/2}\,,$$

and here $1/\mathrm{Area}(V_k(\mathbb{R}^d))$ is the reciprocal of the area of the Stiefel manifold $V_k(\mathbb{R}^d)$ (James, 1954),

$$\frac{1}{\mathrm{Area}(V_k(\mathbb{R}^d))} \;=\; 2^{-k}\prod_{i=1}^{k}\Gamma((d-i+1)/2)\pi^{-(d-i+1)/2}\,.$$

The dependence of $\mathcal{N}_k(d)$ upon k is relatively weak compared to other factors contributing to $\ln p(D|k)$, and so Minka drops $\mathcal{N}_k(d)$ from further consideration in approximating $p(D|k)$. As with the maximum likelihood case (Tipping and Bishop, 1999a), for a fixed choice, k, of the number of principal components, the maximum posterior estimators for $\{\boldsymbol{u}_i\}_{i=1}^k$ are known to be the eigenvectors of $\hat{\boldsymbol{C}}$ corresponding to the k largest eigenvalues of $\hat{\boldsymbol{C}}$. Minka approximates the evidence $p(D|k)$ in Equation (5) using a Laplace approximation, expanding about the maximum posterior solution. The stationary point values of v and $\{l_i\}_{i=1}^k$ are denoted by $\hat{v}$ and $\{\hat{l}_i\}_{i=1}^k$ respectively, and are given by (on taking $\eta\to 0$),

$$\hat{l}_i \;=\; \frac{N\lambda_i}{N-1} \;\simeq\; \lambda_i \quad,\quad \hat{v} \;=\; \frac{N\sum_{j=k+1}^{d}\lambda_j}{(N+1)(d-k)-2}\,. \tag{6}$$

Within this approximation $\hat{l}_i$ provides a point estimate of the i^{th} population covariance eigenvalue Λ_i. For $\alpha < 1$, as we have already commented in the previous section, λ_i can be highly biased and consequently a poor point estimate of Λ_i. Continuing with the Laplace approximation and setting $m = dk - k(k+1)/2$, Minka finds (again after taking $\eta \to 0$),

$$p(D|k) \simeq \frac{1}{\text{Area}(V_k(\mathbb{R}^d))} \left(\prod_{j=1}^{k} \lambda_j \right)^{-N/2} \hat{v}^{-N(d-k)/2} (2\pi)^{(m+k)/2} |\boldsymbol{A}_Z|^{-1/2} N^{-k/2} , \tag{7}$$

where,

$$|\boldsymbol{A}_Z| = \prod_{i=1}^{k} \prod_{j=i+1}^{d} (\hat{\Lambda}_j^{-1} - \hat{\Lambda}_i^{-1})(\lambda_i - \lambda_j) N .$$

The estimator $\hat{\Lambda}_i$ is given by $\hat{\Lambda}_i = \hat{l}_i \simeq \lambda_i$ for $i \leq k$ and $\hat{\Lambda}_i = \hat{v}$ for $i > k$.

Figure 2 shows simulation estimates of the performance of a model selection criterion based upon the evidence given by Equation (7). We have sampled data vectors $\boldsymbol{\xi}_\mu$ from a population covariance $\boldsymbol{C}$ containing three signal components. The noise level has been set to $\sigma^2 = 1$ and the signal strengths are $A_1^2 = 30, A_2^2 = 20, A_3^2 = 10$. The simulation results are averages evaluated over 1000 simulated data sets. Plotted in Fig.2(a) is the probability of selecting the correct model dimension against d, for different fixed values of N. As expected the accuracy of the model selection decreases with increasing d, with greater accuracy for larger sample sizes N at a given value of d. Plotted in Fig.2(b) is the probability of selecting the correct model dimension against d, for different fixed values of α. Note that the smallest value studied, $\alpha = 0.2$, is still greater than the retarded learning transition point of the weakest signal component, which occurs at $\alpha = A_3^{-2} = 0.1$.

The accuracy of the model selection procedure can potentially be improved by noting that PCA can simply be considered as constructing a representation of a matrix, in this case the mean centred sample data matrix. As such the transpose of the representation of the mean centred data matrix is equally as valid, which can be evaluated as the eigen-decomposition of the transpose of the mean centred data matrix. Given that we then model the transposed data matrix using k, N-dimensional vectors rather than k, d-dimensional vectors, then with $N < d$ and thus effectively lower model complexity, we would expect model selection based upon using the transposed mean centred data matrix to display superior accuracy. This is borne out by simulation results for model selection accuracy when applied to the transposed centred data matrix that are also shown in Fig.2. In all cases shown in Fig.2 the accuracy of the model selection is greater when using the transpose of the mean centred data matrix. One should note from Fig.2a, that even with transposing the centred data matrix, the model selection accuracy decreases with increasing data dimensionality d, at fixed sample size N. Taking a data set with $\alpha < 1$ and transposing does not produce an effective value of α that is larger than one - if true this would suggest one could have arbitrarily large effective values of α (by taking $d \to \infty$ at fixed N) and consequently asymptotically perfect model selection even though, as has already been highlighted, the expected spectrum in this limit is indistinguishable from that obtained by sampling from a distribution with an isotropic population covariance matrix. Consequently the accuracy of model selection based upon the sample covariance eigenspectrum will always decrease with increasing d, at fixed N, due to the distorting effects of high data dimensionality. We can attempt to mitigate these effects by taking proper account of the high dimensional contributions to the model evidence. This we do in the next section.

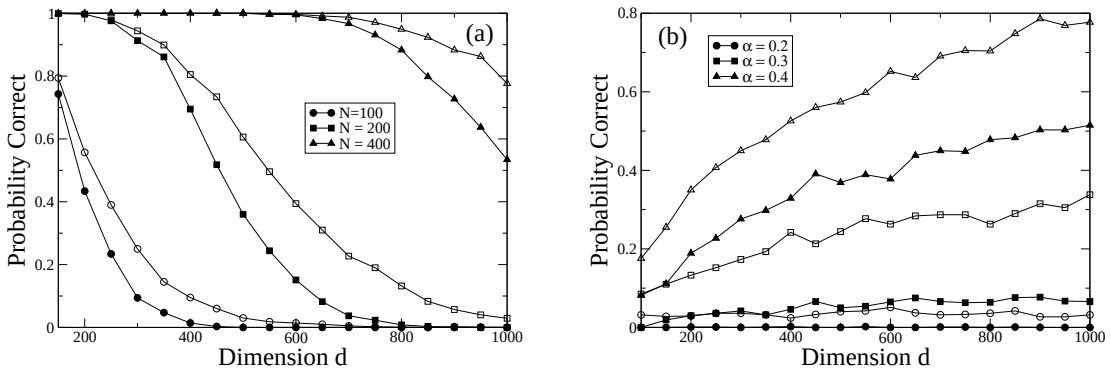


Figure 2: Probability of correct model selection using the method of Minka. The solid lines provide a guide to the eye. (a) & (b) Plots of model selection accuracy against data dimension d - (a) Fixed values of N, (b) Fixed values of α. The data is generated with a population covariance $\boldsymbol{C}$ containing three signal components -see main text for details. Solid symbols represent simulation results from the model selection procedure applied to the mean centred data matrix, whilst open symbols represent simulation results from the model selection procedure applied to the transpose of the mean centred data matrix.

4.2 Overlap Method

Although for $\alpha < 1$ the top k eigenvectors of $\hat{\boldsymbol{C}}$ are the maximum posterior choice of model principal components $\{\boldsymbol{u}_i\}_{i=1}^k$, for non-maximum posterior choices of $\boldsymbol{U}$ one still has a large rotational degeneracy of the k-frame within the d-dimensional space, which will make a large contribution to the integral in Equation (5). The integrand in Equation (5) can be written in terms of the overlaps $R_{ij} = \boldsymbol{u}_i \cdot \boldsymbol{v}_j$ between the model principal components $\boldsymbol{u}_i, i = 1,\ldots,k$, and the eigenvectors $\boldsymbol{v}_j$, $j = 1,\ldots,N-1$, of $\hat{\boldsymbol{C}}$ that correspond to the non-zero eigenvalues of $\hat{\boldsymbol{C}}$. One finds,

$$
\begin{aligned}
& |\boldsymbol{H}\boldsymbol{H}^T + \nu\boldsymbol{I}|^{-(N+1+\eta)/2} \exp(-\frac{N}{2}\mathrm{tr}((\boldsymbol{H}\boldsymbol{H}^T + \nu\boldsymbol{I})^{-1}(\hat{\boldsymbol{C}} + N^{-1}\eta\boldsymbol{I}))) \\
= \;& \exp\left[-\frac{N+1+\eta}{2}\left(\sum_{i=1}^{k} \ln l_i + (d-k)\ln\nu \right) - \frac{N}{2\nu}\sum_{j=1}^{N-1}\lambda_j \right. \\
+ \;& \left. \frac{N}{2}\sum_{i=1}^{k}(\nu^{-1} - l_i^{-1})\sum_{j=1}^{N-1}\lambda_j R_{ij}^2 \; - \; \frac{\eta d}{2\nu} \; + \; \frac{\eta}{2}\sum_{i=1}^{k}(\nu^{-1} - l_i^{-1}) \right] .
\end{aligned}
$$

This suggests performing the integration over $\{\boldsymbol{u}_i\}_{i=1}^k$ in terms of $\{R_{ij}\}$. The volume element that results from integrating over $\{\boldsymbol{u}_i\}_{i=1}^k$ at fixed $\{R_{ij}\}$ is $\det \boldsymbol{M}^{(d-N-1)/2} \times \mathrm{Area}(V_k(\mathbb{R}^{d-N+1}))$, where the matrix elements $M_{ii'} = \delta_{ii'} - \sum_j R_{ij}R_{i'j}$. For high dimensional spaces we might expect the vectors $\boldsymbol{u}_i, \boldsymbol{u}_{i'}$ to be orthogonal over any high-dimensional subspace, not just the entire d-dimensional space. Therefore we can approximate the matrix elements by $M_{ii'} = \delta_{ii'}(1 - \sum_j R_{ij}^2)$, and $\det \boldsymbol{M}$ is easily evaluated. With this approximation the evidence is,

$$p(D|k) \simeq \mathcal{N}_k(d)\frac{\text{Area}(V_k(\mathbb{R}^{d-N+1}))}{\text{Area}(V_k(\mathbb{R}^d))}\int\prod_{ij}dR_{ij}\int\prod_i dl_i\int dv$$
$$\times \quad \exp\left[\frac{d-N-1}{2}\sum_{i=1}^{k}\ln\left(1-\sum_{j=1}^{N-1}R_{ij}^2\right) - \frac{N+1+\eta}{2}\left(\sum_{i=1}^{k}\ln l_i + (d-k)\ln v\right)\right.$$
$$\left. - \frac{N}{2v}\sum_{j=1}^{N-1}\lambda_j + \frac{N}{2}\sum_{i=1}^{k}(v^{-1}-l_i^{-1})\sum_{j=1}^{N-1}\lambda_j R_{ij}^2 - \frac{\eta d}{2v} + \frac{\eta}{2}\sum_{i=1}^{k}(v^{-1}-l_i^{-1})\right] . \qquad (8)$$

Approximations to the model evidence can now be made by approximating this integration over the overlap variables $\{R_{ij}\}$, and consequently this approach is termed the "overlap" method. For large values of d and N we would expect the integral in Equation (8) to be dominated by the stationary points of the exponent and a Laplace approximation to the integral can be constructed. Denoting stationary point values by $\hat{v}, \hat{l}_i, \hat{R}_{ij}$, it is an easy matter to find that, on taking $\eta \to 0$, stationary points of Equation (8) satisfy for some j,

$$1-\hat{R}_{ij}^2 = \frac{(\hat{v}^{-1}-\hat{l}_i^{-1})N\lambda_j}{d-N-1} \quad , \quad \hat{R}_{ij'} = 0 \ , \ j' \neq j \, .$$

The dominant stationary point solution has the overlap between the i^{th} signal direction estimate, $\boldsymbol{u}_i$ and the i^{th} sample covariance eigenvector, $\boldsymbol{v}_i$, being non-zero, that is, $\hat{R}_{ii}^2 > 0$, $\hat{R}_{ii'}^2 = 0, \forall i \neq i'$,

For $j > k$ the dominant stationary point has $\hat{R}_{ij}^2 = 0$. Within this approximation the expectation value of R_{ij}^2 will be $O(N^{-1})$ due to small fluctuations about this stationary point. However, we have an extensive number, that is, proportional to N, of such overlap variables. Thus we expect $\sum_{j>k}R_{ij}^2 \sim 1$, and consequently the contribution from these small fluctuations cannot be ignored. The fluctuations in R_{ij}, for $j > k$, collectively affect the stationary point behaviour of the overlaps R_{ij} for $j \leq k$. To progress we integrate out the fluctuations by setting,

$$b_i = \sum_{j>k} R_{ij}^2 \, ,$$

and perform the integration over $\{R_{ij}\}_{j>k}$ by writing,

$$\int\prod_i\prod_{j>k}dR_{ij} = \int\prod_i\prod_{j>k}dR_{ij}\prod_i db_i\,\delta\left(b_i - \sum_{j>k}R_{ij}^2\right) .$$

Using the standard Fourier representation of a Dirac δ-function,

$$\delta(x) = \frac{1}{2\pi}\int_{-i\infty}^{i\infty}dp\,e^{px} \, ,$$

we obtain,

$$\int\prod_i\prod_{j>k}dR_{ij} = \frac{1}{(2\pi)^k}\int\prod_i db_i\,dp_i\prod_i\prod_{j>k}dR_{ij}\exp\left[\sum_i p_i\left(b_i - \sum_{j>k}R_{ij}^2\right)\right] , \qquad (9)$$

where the path of integration for p_i is between $-i\infty$ and $+i\infty$. Combining the integrand in Equation (9) with the integrand in Equation (8), the integration over $\{R_{ij}\}_{j>k}$ is Gaussian and so easily performed. We obtain,

$$\int dv \int \prod_{i=1}^{k} dl_i db_i dp_i \prod_i \prod_{j\leq k} dR_{ij} \exp\left(\sum_{i=1}^{k} p_i b_i \; + \; \frac{1}{2}(d-N-1)\sum_{i=1}^{k} \ln[1-\sum_{j=1}^{k} R_{ij}^2 - b_i] \right.$$
$$-\frac{1}{2}\sum_{i=1}^{k}\sum_{j>k} \ln[2p_i - N(v^{-1}-l_i^{-1})\lambda_j] \; - \; \frac{N+1}{2}\left[\sum_{i=1}^{k} \ln l_i \; + \; (d-k)\ln v\right]$$
$$\left. -\frac{N}{2}v^{-1}\sum_{j=1}^{N-1}\lambda_j \; + \; \frac{N}{2}\sum_{i=1}^{k}(v^{-1}-l_i^{-1})\sum_{j=1}^{k}\lambda_j R_{ij}^2 \right) . \tag{10}$$

With the path of integration for p_i being along the imaginary axis the remaining integrals in Equation (10) are approximated via steepest descent (Wong, 1989). For brevity we give only the solutions to the saddle point equations, with the caret again denoting saddle-point values of the corresponding integration variables,

$$\hat{v} \;=\; \frac{N}{(N+1)(d-k)}\left[\sum_{j=1}^{N-1}\lambda_j \; - \; \sum_{i=1}^{k}(1+N^{-1})\hat{l}_i\right] , \tag{11}$$

$$0 \;=\; \hat{l}_i^2\hat{v}^{-1}(1+N^{-1}) \; - \; \hat{l}_i(\lambda_i\hat{v}^{-1} - \alpha^{-1} + 1 + N^{-1}(k+3)) \; + \; \lambda_i \, , \tag{12}$$

$$\hat{R}_{ii}^2 \;=\; 1 \; - \; \frac{(d-N-1)}{N(\hat{v}^{-1}-\hat{l}_i^{-1})\lambda_i} \; - \; \frac{1}{N}\sum_{j>k}\frac{1}{(\hat{v}^{-1}-\hat{l}_i^{-1})(\lambda_i-\lambda_j)} \, , \tag{13}$$

$$\hat{R}_{ij}^2 \;=\; 0 \quad , j\neq i \, , \; j\leq k \, ,$$

$$\hat{p}_i \;=\; \frac{N}{2}(\hat{v}^{-1}-\hat{l}_i^{-1})\lambda_i \, , \tag{14}$$

$$\hat{b}_i \;=\; 1 \; - \; \hat{R}_{ii}^2 \; - \; \frac{(d-N-1)}{N(\hat{v}^{-1}-\hat{l}_i^{-1})\lambda_i} \, . \tag{15}$$

Again the saddle-point solution values $\hat{v}$ and $\hat{l}_i$ provide us with point estimates for the population noise level σ^2 and population signal eigenvalue Λ_i respectively. Equations (11) and (12) can be solved efficiently via an iterative process starting from an initial estimate of $\hat{v} = d^{-1}\sum_j \lambda_j$. Obtaining real-valued estimates, $\hat{l}_i$, for the population covariance eigenvalues is clearly dependent upon the quadratic equation in (12) having a non-negative discriminant. In practice, we have interpreted complex-valued estimates $\hat{l}_i$ for a particular choice of signal dimensionality k as meaning that the particular choice for k is not appropriate and should not be considered. From analysis of the asymptotic behaviour of the "overlap" approximation (see next section) we find that the discriminant of Equation (12) becomes negative for sample covariance eigenvalues λ_i which are below the edge of the Marčenko-Pastur bulk distribution given in Equation (2), that is, $\lambda_i < \lambda_{max} = \sigma^2(1+\alpha^{-\frac{1}{2}})^2$, so that indeed a negative discriminant is consistent with attempting to extract more signal components than can be genuinely distinguished from an isotropic population covariance. In other words complex solutions to Equation (12) suggest that the data do not support a model with that number, k, of signal components.

Once solutions for $\hat{v}$ and $\{\hat{l}_i\}_{i=1}^k$ have been obtained, values for $\hat{R}_{ii}^2, \hat{p}_i, \hat{b}_i$ follow from Equations (13), (14) and (15) respectively. Following Minka (2001a) and dropping the relatively weak k-dependence in $\mathcal{N}_k(d)$ we derive an approximation for the log-evidence as,

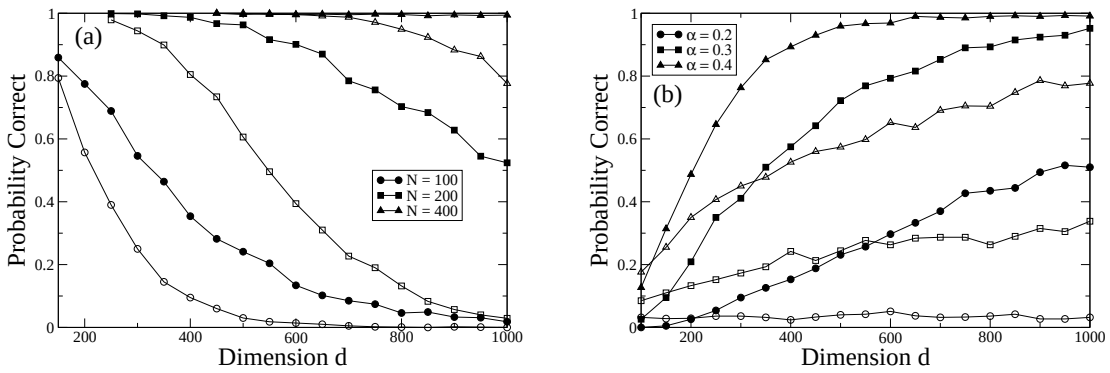


Figure 3: Plot of model selection accuracy for the "overlap" method. (a)Plot of model selection accuracy against data dimension d at fixed values of N. (b)Plot of model selection accuracy against data dimension for fixed values of α. For comparison open symbols represent simulation results from the model selection procedure of Minka applied to the transpose of the mean centred data matrix.

$$\begin{aligned}
\ln p(D|k) \;\simeq\; & \frac{N}{2}\sum_{i=1}^{k}(\hat{v}^{-1}-\hat{l}_i^{-1})\lambda_i \;-\; \frac{k}{2}(d-N-1) \;+\; k\frac{d-N-1}{2}\ln\left(\frac{d-N-1}{N}\right) \\
- \;& \frac{d-N-1}{2}\sum_{i=1}^{k}\ln((\hat{v}^{-1}-\hat{l}_i^{-1})\lambda_i) \;-\; \frac{k}{2}(N-k)\ln N \;-\; \frac{N-k}{2}\sum_{i=1}^{k}\ln(\hat{v}^{-1}-\hat{l}_i^{-1}) \\
- \;& \frac{1}{2}\sum_{i=1}^{k}\sum_{j>k}\ln(\lambda_i-\lambda_j) \;-\; \frac{N+1}{2}\sum_{i=1}^{k}\ln\hat{l}_i \;-\; \frac{N+1}{2}(d-k)\ln\hat{v} \;-\; \frac{N}{2}\hat{v}^{-1}\sum_{j=1}^{N-1}\lambda_j \\
+ \;& \frac{k}{2}(N-k-1)\ln 2\pi \;+\; \ln\left(\frac{\mathrm{Area}(V_k(\mathbb{R}^{d-N+1}))}{\mathrm{Area}(V_k(\mathbb{R}^{d}))}\right) \;-\; \frac{1}{2}\ln\det \boldsymbol{H}_s \\
+ \;& \frac{3k+k^2+1}{2}\ln 2\pi\,, \qquad (16)
\end{aligned}$$

where $\boldsymbol{H}_s$ is the Hessian of the exponent in the integrand evaluated at the saddle point. The last two terms in (16) come from integrating over the small fluctuations about the saddle point. Since the Hessian is of small dimension, and so not strongly dependent on N and d, we subsequently drop the last two terms from our approximation of the log-evidence. The "overlap" approximation to the log-evidence, given in Equation (16), can be used for model selection by selecting the value of k that has the highest value of $\ln p(D|k)$.

Figure 3 shows simulation estimates of the accuracy of the "overlap" model selection criterion given in Equation (16). Fig.3(a) shows the probability of selecting the correct model dimension against d, for different fixed values of N. Plotted in Fig.3(b) is the probability of selecting the correct model dimension against d, for different fixed values of α. Sample sizes and model pa-

rameter values are identical to those in Figure 2. Also reproduced (open symbols) in Fig.3(a) and Fig.3(b) are the simulation estimates of model selection accuracy for Minka's approximation to the model evidence applied to the transposed mean centred data. From Fig.3(a) it is clear that the "overlap" model selection criterion only suffers from degradation in performance at significantly higher values of dimension d compared to the approximation to the evidence in Equation (7). Similarly, Fig.3(b) demonstrates the superior model selection accuracy of the "overlap" method for increasing dimensionality d, at fixed values of α.

5. Asymptotic Analysis

The "overlap" approximation to the model evidence has been developed by applying a steepest descent approximation to the Bayesian evidence that has been re-formulated in terms of integration over variables that remain finite in number in the distinguished asymptotic limit $d, N \to \infty$, at fixed α. The "overlap" approximation essentially contains the leading order term of an asymptotic expansion of the evidence in that distinguished limit. It would be expected that the approximation to the model evidence would therefore become increasingly accurate in this limit. Note that this is very different from the traditional large sample limit $N \to \infty$ at fixed d, for which Minka's approximation to the Bayesian evidence will become increasingly accurate. It has been argued that since for many real high-dimensional data sets $\alpha \ll 1$, one would expect that approximations to the model evidence that are accurate in the distinguished limit will have superior model selection accuracy at finite values of d, N. The simulation results presented in Fig.3 would appear to confirm this. However, more concrete understanding of the accuracy of the "overlap" method in the distinguished asymptotic limit is required. A theoretical analysis of model selection accuracy in this limit would provide us with a firmer comparison of Minka's original Laplace approximation and the "overlap" method, in addition to the comparison provided by simulation study in Section 4.2. A number of quantities such as the eigenvalue spectrum are self-averaging in the asymptotic limit, that is, have vanishing sampling variation, so that for large data dimensions, d, the value for a single data set, $\{\boldsymbol{\xi}_\mu\}$, is well approximated by the ensemble average over data sets. Studying the ensemble expectation, in the asymptotic limit of $d \to \infty$ at fixed α, of the "overlap" approximation to the model evidence provides us with insight into its accuracy as a model selection procedure for high dimensional data.

From Equation (11) it is evident that $\hat{v} = d^{-1} \sum_{j=1}^{N-1} \lambda_j + O(N^{-1})$ as $N \to \infty$. Consequently, due to the self-averaging nature of the sample covariance eigenvalue spectrum, we have that $\hat{v} \to \mathrm{E}_{\boldsymbol{\xi}}(\lambda)$ as $N \to \infty$, where we have used $\mathrm{E}_{\boldsymbol{\xi}}(\cdot)$ to denote expectation over the ensemble of sample data sets. We already commented in Section 3 that $\mathrm{E}_{\boldsymbol{\xi}}(\lambda) = \sigma^2$ in the asymptotic limit $N \to \infty$ at fixed α, and so $\hat{v}$ provides an asymptotically unbiased estimate of the population noise level. Estimates of the population signal eigenvalues are given by $\{\hat{l}_i\}_{i=1}^k$, and in the distinguished asymptotic limit solutions to Equation (12) for $\hat{l}_i$ are given by,

$$\hat{l}_i = \frac{\hat{v}}{2}\left[(1+\lambda_i\hat{v}^{-1}-\alpha^{-1}) \pm \sqrt{(1+\lambda_i\hat{v}^{-1}-\alpha^{-1})^2 - 4\lambda_i\hat{v}^{-1}}\right] . \tag{17}$$

If we consider a "spiked" population covariance model of the form in Equation (1) the population covariance eigenvalues correspond to signal eigenvalues $\Lambda_i = \sigma^2(1+A_i)$, $i \leq S$ and noise eigenvalues $\Lambda_i = \sigma^2$, $i > S$. The resulting expected sample covariance eigenspectrum is given in Equation (3). Taking those sample eigenvalues which are separated from the bulk and also those at the upper bulk edge and substituting into Equation (17) we obtain on setting $\hat{v} = \sigma^2$ (on taking the positive

solution branch),

$$\begin{aligned}\hat{l}_i &= \sigma^2(1+A_i) \quad ,\text{for } \lambda_i = \sigma^2(1+A_i)(1+(1/\alpha A_i))\,,\\ \hat{l}_i &= \sigma^2(1+\alpha^{-\frac{1}{2}}) \quad ,\text{for } \lambda_i = \sigma^2(1+\alpha^{-\frac{1}{2}})^2\,.\end{aligned}$$

For sample covariance eigenvalues that are below the edge of the Marčenko-Pastur bulk distribution, that is, $\lambda_i < \sigma^2(1+\alpha^{-\frac{1}{2}})^2$, we obtain only complex solutions from Equation (17). Conversely, when $\lambda_i = \sigma^2(1+A_i)(1+(1/\alpha A_i))$, that is, when the sample covariance spectrum displays eigenvalues which are distinct from the bulk of the distribution, the estimator $\hat{l}_i = \sigma^2(1+A_i) = \Lambda_i$ and so gives an asymptotically unbiased estimate of the population signal eigenvalue Λ_i.

What is the asymptotic behaviour of the log-evidence? Inspecting Equation (16) we can see that, potentially, we need to evaluate $O(N^{-1})$ contributions to $\mathrm{E}_{\boldsymbol{\xi}}(\hat{\nu})$. However, it is easily shown that $O(N^{-1})$ contributions to $\mathrm{E}_{\boldsymbol{\xi}}(\hat{\nu})$ cancel out when evaluating $\mathrm{E}_{\boldsymbol{\xi}}(\ln p(D|k))$, and so we do not pursue them further here. We can evaluate the ensemble average $\mathrm{E}_{\boldsymbol{\xi}}(\sum_{j>k}\ln(\lambda_i-\lambda_j))$ through use of the replica trick (see Appendix A). Specifically we have for $\alpha > A_i^{-2}$,

$$\lim_{N,d\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right) \;=\; \ln\sigma^2 \;-\; (\alpha^{-1}-1)\ln(1+A_i) \;+\; \alpha^{-1}\ln A_i \;+\; \frac{1}{\alpha A_i}\,, \qquad (18)$$

whilst for $\alpha < A_i^{-2}$ we have,

$$\lim_{N,d\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right) \;=\; \ln\sigma^2 \;-\; (\alpha^{-1}-1)\ln(1+\alpha^{-\frac{1}{2}}) \;+\; \alpha^{-1}\ln\alpha^{-\frac{1}{2}} \;+\; \alpha^{-\frac{1}{2}}\,. \quad (19)$$

The asymptotic behaviour of the ratio $\mathrm{Area}(V_k(\mathbb{R}^{d-N+1}))/\mathrm{Area}(V_k(\mathbb{R}^d))$ is easily evaluated to give,

$$\ln\left(\frac{\mathrm{Area}(V_k(\mathbb{R}^{d-N+1}))}{\mathrm{Area}(V_k(\mathbb{R}^d))}\right) \;=\; \frac{Nk}{2}\left[-\ln\pi \;+\; \ln\frac{d}{2} \;-\; (\alpha^{-1}-1)\ln(1-\alpha) \;-1\right] \;+\; O(\ln N)\,. \quad (20)$$

Substituting Equations (18),(19),(20) and the asymptotic values for $\hat{\nu}$ and $\hat{l}_i$ into Equation (16), we obtain after some straight-forward algebra,

$$\begin{aligned}\mathrm{E}_{\boldsymbol{\xi}}(\ln p(D|k)) &= \frac{N}{2}\sum_{i=1}^{k}\Theta(\alpha-A_i^{-2})\left[A_i-\frac{1}{\alpha A_i} \;+\; (\alpha^{-1}-1)\ln\left(\frac{1+A_i}{1+(1/\alpha A_i)}\right) \;+\; \alpha^{-1}\ln\left(\frac{1}{\alpha A_i^2}\right)\right]\\ &- \frac{Nd}{2}\ln\sigma^2 \;-\; \frac{Nd}{2} \;+\; O(\ln N)\,. \qquad (21)\end{aligned}$$

If we set $x = \alpha^{-1}$ we can write the summand in Equation (21) as $\Theta(\alpha - A_i^{-2})\,f(x,A_i)$ where,

$$f(x,A) = A \;+\; x\ln x \;+\; (x-1)\ln(1+A) \;-\; 2x\ln A \;-\; (x-1)\ln(1+xA^{-1}) \;-\; xA^{-1}\,.$$

We find that,

$$f(A^2,A) \;=\; 0 \quad , \quad \left.\frac{\partial f}{\partial x}\right|_{x=A^2} \;=\; 0 \quad , \quad \frac{\partial^2 f}{\partial x^2} \;>\; 0 \;\text{ for } x \;<\; A^2\,,$$

and so for $\alpha > A_i^{-2}$ the summand in Equation (21) is positive. Consequently if $\alpha > A_i^{-2}$, so that the sample covariance eigenvalue spectrum reflects the presence of the signal $\boldsymbol{B}_i$, then the addition of

i^{th} principal component results in an increase in the asymptotic approximation to the log-evidence. Conversely if $\alpha < A_i^{-2}$ there is no change in the asymptotic approximation to the log-evidence on including the i^{th} principal component. This is a satisfying result since we have already commented in Section 3 that for $\alpha < A_i^{-2}$ the sample covariance eigenspectrum is asymptotically indistinguishable from that produced from a population model with $A_i \equiv 0$, and so therefore from a Bayesian model selection perspective all population models with $A_i < \alpha^{-\frac{1}{2}}$ are equally likely (provide an equally accurate description of the observed data). Ultimately this is due to the fact that we are considering models with a finite number, k, of signal components, and so in the asymptotic limit we are considering a vanishingly small proportion of sample covariance eigenvalues as representing signal components. With the non-zero sample covariance eigenvalues giving a dense covering of the range $[\lambda_{min}, \lambda_{max}]$ in the asymptotic limit, the largest few sample covariance eigenvalues, which are not distinct from the Marčenko-Pastur bulk distribution given in Equation (2) will be aggregated at the upper edge of the bulk, where they do not lead to any change in the log-evidence. For finite sample sizes we would expect the higher order terms in the expansion of the log-evidence to lead to a decrease in the log-evidence on inclusion of principal components that correspond to sample covariance eigenvalues that are below the bulk edge. However, in the asymptotic limit we can apply an Occam's Razor like argument and only select those principal components that increase the log-evidence. The limiting model selection estimate, $\hat{S}$, for the true signal dimensionality, S, then simply corresponds to counting the number of sample covariance eigenvalues that are beyond the upper edge of the Marčenko-Pastur bulk distribution. That is,

$$\hat{S} \;=\; \sum_{j=1}^{d} \Theta(\lambda_j \;-\; \lambda_{max}) \,.$$

The asymptotic analysis of the "overlap" method reveals that unbiased estimates of the population signal eigenvalues can be recovered and that, asymptotically, model selection based upon the "overlap" approximation to the log-evidence performs optimally. From Fig.2b it would appear that, at least for larger values of α, model selection based upon Minka's approximation to the log-evidence also approaches 100% accuracy as $d \to \infty$. Is it possible that the two different approximations to the log-evidence asymptotically have the same model selection performance? Starting from Minka's approximation to the Bayesian evidence $p(D|k)$ given in Equation (7) we have,

$$\begin{aligned} \ln p(D|k) \;\;\simeq\;\; & -\ln \text{Area}(V_k(\mathbb{R}^d)) \;-\; \frac{N}{2}\sum_{i=1}^{k} \ln \lambda_i \;-\; \frac{N}{2}(d-k)\ln \hat{\nu} \;+\; \frac{m+k}{2}\ln 2\pi \\ & -\frac{1}{2}\sum_{i=1}^{k}\sum_{j=i+1}^{d}\Big[\ln(\hat{\Lambda}_j^{-1} - \hat{\Lambda}_i^{-1}) \;+\; \ln(\lambda_i - \lambda_j) \;+\; \ln N\Big] \;-\; \frac{k}{2}\ln N \,, \end{aligned} \tag{22}$$

where $\hat{\Lambda}_i = N\lambda_i/(N-1)$ for $i \leq k$ and $\hat{\Lambda}_i = \hat{\nu}$ for $i > k$, with $\hat{\nu}$ defined in Equation (6). In this instance $O(N^{-1})$ contributions to $\text{E}_{\boldsymbol{\xi}}(\hat{\nu})$ do make a contribution to the leading order asymptotic term in $\text{E}_{\boldsymbol{\xi}}(\ln p(D|k))$. From the definition of the point estimate $\hat{\nu}$ in (6) we find,

$$\text{E}_{\boldsymbol{\xi}}(\hat{\nu}) \;=\; (1 + N^{-1}(k\alpha - 1))\text{E}_{\boldsymbol{\xi}}(d^{-1}\text{tr}\hat{\boldsymbol{C}}) \;-\; \frac{\alpha}{N}\sum_{j=1}^{k}\text{E}_{\boldsymbol{\xi}}(\lambda_j) \;+\; O(N^{-2}) \,.$$

For the "spiked" covariance model of Equation (1) this can then be refined to,

$$\begin{aligned}\mathrm{E}_{\boldsymbol{\xi}}(\hat{v}) &= \sigma^2 + \frac{\alpha\sigma^2}{N}\sum_{j=1}^{S} A_j + \frac{\sigma^2}{N}(k\alpha - 1) \\ &\quad - \frac{\alpha\sigma^2}{N}\sum_{i=1}^{k}\left[\Theta(\alpha - A_i^{-2})(1+A_i)(1+(1/\alpha A_i)) + \Theta(A_i^{-2}-\alpha)(1+\alpha^{-\frac{1}{2}})^2\right] \\ &\quad + O(N^{-2})\,.\end{aligned}$$

Retaining only k-dependent terms, the leading order asymptotic contribution to $\mathrm{E}_{\boldsymbol{\xi}}(\ln p(D|k))$ can be obtained within this approximation as,

$$\mathrm{E}_{\boldsymbol{\xi}}(\ln p(D|k)) = \frac{N}{2}\sum_{i=1}^{k}\left[\Theta(\alpha - A_i^{-2})\, f_M(x,A_i) + \Theta(A_i^{-2}-\alpha)\, f_M(x,\alpha^{-\frac{1}{2}})\right] + O(\ln N)\,,$$

where the subscript M on the function $f_M(x,A)$ is used to denote the asymptotic incremental change to the log-evidence obtained from Minka's approximation given in Equation (22), and again $x = \alpha^{-1}$. Specifically $f_M(x,A)$ is given as,

$$f_M(x,A) = A + x\ln x + (x-1)\ln(1+A) - 2x\ln A - x\ln\left[1 + xA^{-1} + xA^{-2}\right]\,. \qquad (23)$$

The transition point at which a signal component is strong enough to be distinguishable from the Marčenko-Pastur bulk distribution in Equation (2) is given by a signal strength $A = \alpha^{-\frac{1}{2}}$. If we put $A = y\alpha^{-\frac{1}{2}} = y\sqrt{x}$, then y directly measures the signal strength relative to that at which it is first detectable. We can then write Equation (23) as,

$$f_M(x, A = y\sqrt{x}) = (x-1)\ln(1+y\sqrt{x}) - x\ln\left(1 + y\sqrt{x} + y^2\right) + y\sqrt{x}\,.$$

A plot of $f_M(x = \alpha^{-1}, A = y\alpha^{-\frac{1}{2}})$ against y for different fixed values of α is shown in Figure 4. From Fig.4 we can see that at the transition point, $y = 1$, f_M is negative, and so selection of the i^{th} principal component will result in a reduction of the log-evidence, even if the signal strength A_i is sufficiently strong enough for the i^{th} sample covariance eigenvalue to be distinct from the Marčenko-Pastur bulk distribution. Thus, even though a detectable signal is present model selection based upon Equation (22) would not include that signal component. For the largest value of α shown f_M does not become positive until approximately $y > 1.8$. Therefore, even for $\alpha = 0.9$, not until the signal strength A_i is 1.8 times stronger than it need be for detection will the i^{th} signal component be correctly selected whilst using Minka's approximation to the log-evidence in Equation (22). For smaller values of α even stronger signal strengths are required, for example, $y > 2.0$ at $\alpha = 0.1$. For the simulations results shown in Fig.2b it is only at the largest value of α shown that we have $f_M > 0$ for all three signal components, and thus that all three signal components are guaranteed to be detectable in the asymptotic limit.

6. Comparison with Frequentist Approaches

In the distinguished asymptotic limit $N, d \to \infty$ the model selection process based upon the "overlap-method"approximation to the log-evidence simplifies (after applying a Occam's Razor like argument) to retaining those principal components whose corresponding eigenvalues are greater than

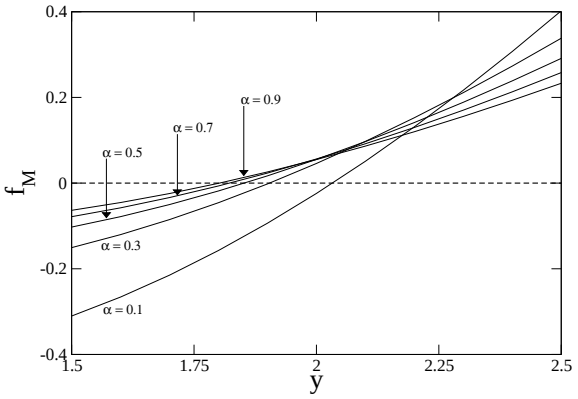


Figure 4: Plot of the function $f_M(x=\alpha^{-1}, A=y\alpha^{-\frac{1}{2}})$ against y for different values of α. $N f_M/2$ represents the incremental change (to leading order) in the log-evidence on retaining a principal component corresponding to a signal component of strength $A=y\alpha^{-\frac{1}{2}}$. The horizontal dashed line denotes the zero level for f_M.

the upper spectral edge, $\lambda_{max}=\sigma^2(1+\alpha^{-\frac{1}{2}})^2$, of the bulk eigenvalue distribution. Whilst this result appears intuitive from the viewpoint of the behaviour of eigenspectra of large sample covariance matrices presented in Section 3, we have also shown that not all approximations to the Bayesian evidence reduce in the asymptotic limit to this optimal choice for model selection. How then does the "overlap" method for model selection compare to other approaches, for example more traditional non-Bayesian approaches for dimensionality selection in PCA? In the asymptotic limit $N\rightarrow\infty$, where we have an infinite amount of data, we would naively expect frequentist and correctly formulated Bayesian approaches to model selection to give similar answers.

One of the most commonly applied techniques for dimensionality selection for PCA is to select sample covariance eigenvalues (and corresponding eigenvectors) that account for a fixed percentage of the total variance, for example, 90%. Typically this may only be the top two or three eigenvalues. Alternative methods consist of producing a 'scree plot', that is, plot of eigenvalue against rank, and attempting to detect by eye an 'elbow' in the plot where there is a significant change in scale of the sample covariance eigenvalues, supposedly reflecting the change from signal dominated eigenvalues to noise dominated eigenvalues. However, with sample covariance eigenvalues potentially being highly biased even when the population covariance is isotropic this is not always a reliable or easily implemented technique.

Hypothesis tests have been developed to detect departure from sphericity of the population covariance, based upon using $\mathrm{tr}\hat{\boldsymbol{C}}$ as the test statistic (John, 1971; Nagao, 1973). This approach has been modified by Ledoit and Wolf (2002) to account for smaller sample sizes but is still essentially only appropriate for $\alpha>1$. The effect of smaller values of α can be accounted for since the asymptotic form of the expected spectrum is given by the Marčenko-Pastur distribution (2) when $\boldsymbol{C}=\sigma^2\boldsymbol{I}$. Wachter has used this by producing Q-Q plots of the sample covariance eigenvalue quantiles against the Marčenko-Pastur distribution quantiles (Wachter, 1976). Sample covariance eigenvalues above

the 45 degree line in these Wachter plots indicate potentially signal containing principal components. At finite values of d a more principled, but non-Bayesian, approach would be to perform a series of iterative hypothesis tests whereby the null-hypothesis H_0 is that of a model containing k signal components. Comparison of the $(k+1)^{\text{th}}$ sample covariance eigenvalue, λ_{k+1}, against the sampling distribution of λ_{k+1} under H_0 would allow for potential rejection of the null-hypothesis and inclusion of the $(k+1)^{\text{th}}$ principal component as representing genuine signal in the data. After setting a rate at which one wishes to control the Type-I error, for example, $\gamma = 0.05$, testing of the $(k+1)^{\text{th}}, (k+2)^{\text{th}}, \ldots$ principal components proceed via,

$H_0 \; : \; \boldsymbol{C} \equiv \hat{\sigma}^2 \boldsymbol{I} \;\; , \;\; \hat{\sigma}^2 \; = \; d^{-1} \sum_{j=1}^{d} \lambda_j \;\; , \;\; k = 0$
while $p(\lambda > \lambda_{k+1} | k, d, N) < \gamma$
$k \rightarrow k+1$
$\hat{\Lambda}_k \; = \; \lambda_k$
$\hat{\sigma}^2 \; = \; (d-k)^{-1} \sum_{j=k+1}^{d} \lambda_j$
$H_0 \; : \; \boldsymbol{C} \equiv \text{diag}(\hat{\Lambda}_1, \ldots, \hat{\Lambda}_k, \hat{\sigma}^2, \ldots, \hat{\sigma}^2)$
end while

To implement this testing procedure we need the cumulative sampling distribution $p(\lambda > \lambda_{k+1} | k, d, N)$ of the $(k+1)^{\text{th}}$ sample covariance eigenvalue under the null hypothesis of $\boldsymbol{C}$ containing k signal components - that is the probability, when the population covariance contains only k signal components, of the $(k+1)^{\text{th}}$ sample covariance eigenvalue being larger than the eigenvalue λ_{k+1} observed in the real sample data. Johnstone (2001) has derived the sampling distribution for $k = 0$ by extending the analysis of Tracy and Widom (1996) on the Gaussian Orthogonal Ensemble (GOE) of random matrices. We can define location and scale constants,

$$\mu_{Nd} \; = \; N^{-1} \left(\sqrt{N-1} \, + \, \sqrt{d} \right)^2 \, ,$$

and

$$\sigma_{Nd} \; = \; N^{-1} \left(\sqrt{N-1} \, + \, \sqrt{d} \right) \left(\frac{1}{\sqrt{N-1}} \, + \, \frac{1}{\sqrt{d}} \right)^{\frac{1}{3}} \, .$$

Then for data drawn from an isotropic population covariance, $\boldsymbol{C} \; = \; \sigma^2 \boldsymbol{I}$, the largest sample covariance eigenvalue λ_1 (suitably centred and scaled) converges in distribution to the Tracy-Widom distribution W_1. Specifically one has,

$$\frac{(\lambda_1 / \sigma^2) - \mu_{Nd}}{\sigma_{Nd}} \xrightarrow{\mathcal{D}} W_1 \sim F_1 \, ,$$

where,

$$F_1(s) \; = \; \exp \left\{ -\frac{1}{2} \int_s^{\infty} q(x) + (x-s) q^2(x) dx \right\} \, ,$$

with $q(x)$ being the solution to the Painlevé II differential equation that is asymptotically equivalent to the Airy function $\text{Ai}(x)$,

$$\begin{aligned} \frac{d^2 q(x)}{dx^2} \; &= \; x q(x) \, + \, 2 q^3(x) \, , \\ q(x) \; &\sim \; \text{Ai}(x) \;\; , x \rightarrow \infty \, . \end{aligned}$$

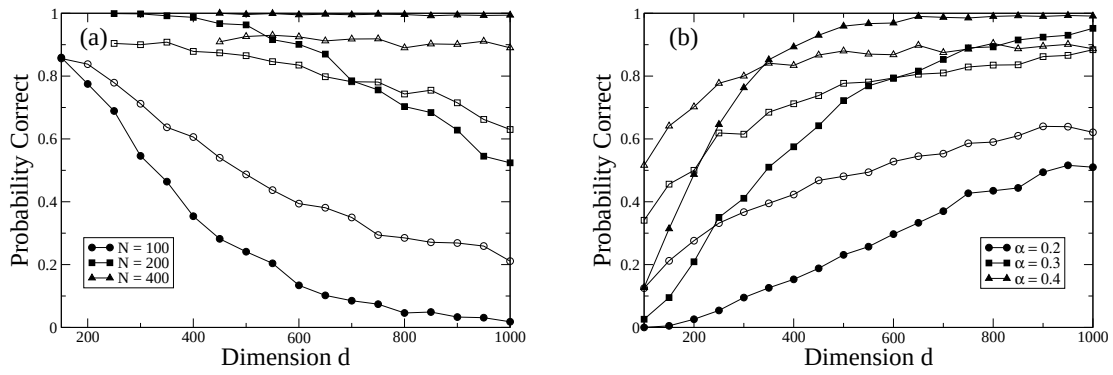


Figure 5: Comparison of the model selection accuracy for the "overlap" method (solid black symbols) with a null hypothesis test based upon the Tracy-Widom distribution for the largest eigenvalue of a sample covariance matrix (open symbols). a)Plot of model selection accuracy against data dimension d at fixed values of N. (b)Plot of model selection accuracy against data dimension for fixed values of α.

Note that the centering constant $\mu_{Nd} \to \lambda_{max}$, up to an irrelevant factor of σ^2. That is, the edge of the Marčenko-Pastur distribution, as $N, d \to \infty$ at fixed α. More recent analysis of the distribution of λ_1 when the data is complex and contains signal has been performed by Baik et al. (2005). The authors provide conjectures for the behaviour of the sampling distribution of λ_1 when the data is real, based upon their analysis of the complex case, but this still does not provide a means of calculating the sampling distribution $p(\lambda_{k+1}|k,d,N)$ for $k > 0$. Instead Johnstone (2001) derives the inequality $p(\lambda > \lambda_{k+1}|k,d,N) < p(\lambda > \lambda_1|0,d-k,N)$, with the latter distribution being given in terms of the Tracy-Widom distribution. Consequently, at finite N this provides a conservative hypothesis test since use of $p(\lambda_1|0,d-k,N)$ yields an over-estimate of the tail area of the sampling distribution $p(\lambda_{k+1}|k,d,N)$, and therefore an over-estimate of the Type-I error rate. With the variance σ^2_{Nd} tending to zero as $N \to \infty$, then in the asymptotic limit $N \to \infty$ the series of hypothesis tests given above corresponds simply to determining how many sample covariance eigenvalues λ_i are above λ_{max} - the edge of the Marčenko-Pastur bulk distribution - and so, as naively expected, is in agreement with the behaviour of the "overlap" method in the distinguished asymptotic limit.

Although in the distinguished asymptotic limit the Bayesian and frequentist approaches to model selection agree, it is interesting to compare model selection accuracies for finite values of N and d. For real data sets the sampling distribution of the individually ranked eigenvalues will have an effect upon the performance of the hypothesis test approach and likewise accuracy of point estimates for model parameters will impact upon the performance of the Bayesian methods. Figure 5 shows the model selection accuracy for the "overlap" method compared to that for the null hypothesis test outlined above that is based upon the Tracy-Widom distribution. Fig.5(a) shows the probability of selecting the correct model dimension against d, for different fixed values of N. Plotted in Fig.5(b) is the probability of selecting the correct model dimension against d, for different fixed values of α. Sample sizes and model parameter values are identical to those in Figure 2

and Figure 3. We have controlled the Type-I error at the 5% level ($\gamma = 0.05$) with the value of the abscissa for the 95^{th} centile of the Tracy-Widom distribution taken from Johnstone (2001). Within Fig.5(b) we would expect all model selection accuracies to converge to 1 as $d, N \to \infty$ since all the signal strengths have been chosen to be above their respective retarded learning transition points and therefore the sample eigenvalues corresponding to signal directions are all distinguishable from the Marčenko-Pastur bulk in this limit. However, for finite d and N Fig.5(a) and (b) reveal that overall the hypothesis testing approach has a superior model selection accuracy when both N and α are relatively small. By definition, the hypothesis test only considers a sample covariance eigenvalue to represent a signal if it exceeds that expected from the null model by more than reasonable sampling variation. As sampling variation will be greater at smaller values of N we might expect the hypothesis testing approach to be more sensitive for model selection than the "overlap" approach within this regime, particularly since higher order terms in the asymptotic expansion of the Bayesian evidence, that have not been incorporated into the "overlap" evidence approximation, will be more significant for smaller values of d and N. For larger values of d and N, the conservative nature of any hypothesis testing approach may adversely affect its model selection accuracy in comparison to a Bayesian evidence based approach.

7. Discussion & Conclusions

For calculations within high-dimensional inference problems we have argued that, rather than using results obtained by considering the traditional large sample limit $N \to \infty$, better approximations may be obtained by considering them to be close to the asymptotic value obtained in some distinguished limit, even though the sample size N may naively be considered large enough for routine application of the Laplace approximation to be accurate. What constitutes a large sample size, N, should clearly be defined with respect to the data dimensionality d. For PCA the appropriate distinguished asymptotic limit is $d, N \to \infty$, with $\alpha = N/d$ fixed, though for other models different distinguished limits may need to be considered in order to observe meaningful non-trivial behaviour that is distinct from the large sample limit, $N \to \infty$. For example, statistical physics studies of independent component analysis (ICA) suggests that $d \to \infty$ with $N = \alpha d^{\frac{3}{2}}$ at fixed α would be the appropriate distinguished limit to consider (Urbanczik, 2003). However, irrespective of the particular distinguished limit considered when developing an asymptotic approximation, one needs to be careful to keep track of the increasing number of contributions as $d \to \infty$, and potentially large contributions resulting from the rotational degeneracy of the integrand in the formulation of the Bayesian evidence.

The effect of high data dimensionality on model selection accuracy when $\alpha < 1$ is apparent from the simulation results shown in Fig.2a and Fig.3a. Ultimately this is due to the biased sample covariance eigenvalues and the poor accuracy of the sample covariance eigenvectors in representing the signal directions when $\alpha < 1$. The high-dimensional nature of the data leads to high-dimensional integral formulations of the Bayesian evidence. Approximation of the evidence has to be done carefully. Within the "overlap" method, inclusion of large contributions to the evidence from rotational degeneracy of the model $k-$frame and extensive Gaussian fluctuations leads to improved model selection accuracy. The observation that reformulating the integrand can lead to improved Laplace estimates of marginal distributions is not necessarily a new one (MacKay, 1998). For high-dimensional data the reformulation is essential, and for the "overlap" method reformulation of the evidence calculation in terms of a finite number of variables has ultimately led to an integrand that is better approximated by a single Gaussian, via a steepest descent calculation. There may exist

potentially superior estimation schemes, based upon a Gaussian parametrization of the integrand, that perform well when the integrand is essentially unimodal, for example expectation propagation based schemes (Minka, 2001b) or variational approximation similar to that employed by Bishop (1999b) for model selection within Bayesian PCA, although it should be noted that Bishop (1999b) does not impose an orthogonal constraint upon the low dimensional decomposition of the population covariance. However, it is the fact that one has to reformulate the evidence calculation for a Gaussian approximation to be accurate that is our main finding here, not the particular choice of approximation scheme that one employs once the reformulation has been made. Of greater interest perhaps is the fact that we have been able to demonstrate the asymptotic equivalence of the Bayesian evidence based model selection criterion and the frequentist hypothesis testing approach to model selection. Furthermore, analysis of the asymptotic behaviour of the "overlap" approximation to the log-evidence reveals that the estimators of the population signal eigenvalues are unbiased, at least for the "spiked" covariance models considered here.

The influence of high data dimensionality on estimates of model parameters can be explicitly demonstrated by re-visiting Minka's original Laplace approximation to the evidence. Although Minka's derivation provides a poorer approximation to the model evidence, in the distinguished limit $N, d \to \infty$ at fixed α, in comparison to the "overlap" approximation, it is still the correct leading order approximation in the asymptotic limit $N \to \infty$ at arbitrary fixed values of d. Therefore it contains information about how the model evidence behaves for large values of N and d. This suggests that Minka's Laplace approximation to the model evidence could be re-used to develop improved point estimates of population covariance eigenvalues $\{\Lambda_i\}$. One proceeds by noting that the eigenvectors of $\hat{\boldsymbol{C}}$ are the maximum posterior estimates of $\boldsymbol{U}$ for arbitrary choices of $\{l_i\}$ and v, since projection of the sample data onto the sample covariance eigenvectors retains the greatest variance. We can simply re-use Minka's approach to perform the Gaussian integration over $\boldsymbol{U}$ about this maximum posterior point, yielding $p(D|\{l_i\}, v)$ which can then be optimized with respect to $\{l_i\}$ and v. Specifically we have the following approximation to the log-evidence (taking $\eta \to 0$),

$$\begin{aligned} & -\frac{N+1}{2}\left(\sum_{i=1}^{k}\ln l_i \,+\, (d-k)\ln v\right) \,-\, \frac{N}{2v}\sum_{j=1}^{N-1}\lambda_j \,+\, \frac{N}{2}\sum_{i=1}^{k}(v^{-1}-l_i^{-1})\lambda_i \\ - \quad & \frac{1}{2}\ln|\boldsymbol{A}_Z| \,+\, \ln \mathcal{N}_k(d) \,-\, \ln \mathrm{Area}(V_k(\mathbb{R}^d)) \,+\, \frac{m+k}{2}\ln 2\pi \,-\, \frac{k}{2}\ln N\,, \end{aligned} \tag{24}$$

$$\ln|\boldsymbol{A}_Z| \,=\, m\ln N \,+\, \sum_{i=1}^{k}\left((d-k)\ln(v^{-1}-l_i^{-1}) \,+\, \sum_{j=i+1}^{k}\ln(l_j^{-1}-l_i^{-1}) \,+\, \sum_{j=i+1}^{d}\ln(\lambda_i-\lambda_j)\right).$$

It should be noted that the contribution from $\ln|\mathbf{A}_Z|$ is extensive in d and therefore affects the construction of point estimates for $\{l_i\}$ and v. Retaining only extensive terms in Equation (24) and locating stationary points with respect to l_i and v yields estimators $\hat{l}_i, \hat{v}$. In the asymptotic limit $N \to \infty$ these estimators are given by,

$$\begin{aligned} 0 \;&=\; \hat{v}^{-1}\hat{l}_i^2 \,-\, \hat{l}_i(1+\hat{v}^{-1}\lambda_i-\alpha^{-1}) \,+\, \lambda_i\,, \\ \hat{v} \;&=\; d^{-1}\sum_{j=1}^{N-1}\lambda_j\,. \end{aligned}$$

The equation above, determining the asymptotic behaviour of the estimator $\hat{l}_i$ is asymptotically identical to that given in Equation (12) for the "overlap" method and so, as already noted, gives asymptotically unbiased estimates for the population signal eigenvalue. Although this leading order approximation to the log-evidence can yield asymptotically unbiased estimators of the population covariance eigenvalues, it is still not an accurate estimation of the log-evidence and will still give inferior model selection performance in comparison to the "overlap" method. This is because higher order terms in the asymptotic expansion of the integral in Equation (5) will also be extensive in N, on taking the asymptotic limit $N, d \to \infty$ at fixed α. Ultimately this can seen from the "overlap" reformulation of the integral defining the evidence, which introduces higher than quadratic order terms in the extensive integration variables R_{ij} in the exponent of the integrand in Equation (8). These higher than quadratic order terms only arise for $\alpha < 1$ due to the contribution of the determinant $\det \boldsymbol{M}^{(d-N-1)/2}$ that results on changing integration variables from orthonormal vectors $\{\boldsymbol{u}_i\}_{i=1}^k$ of the model k-frame to the overlap variables R_{ij}. However, as we have demonstrated with the increased model selection accuracy of the "overlap" method, it is important to explicitly reformulate the integration in terms of variables that are finite in number even in the asymptotic limit $N, d \to \infty$ at fixed α.

For the simulations presented within this paper we have taken the signal dimensionality k to be finite and relatively small, for example, $k = 1, 2, 3$, so that $k \ll N < d$. This choice reflects the current interest in "spiked" covariance models and the generic challenge of identifying a fixed low-dimensional subspace as more and more features are considered. However, it is entirely feasible to imagine scenarios where the signal dimensionality is much larger than $k = 3$, and potentially even comparable to the sample size N. The derivation of the approximation to the log-evidence given in Equation (16) is valid for any finite value of k and thus can be used for model selection even for data sets where larger values of k are appropriate. Studying the accuracy of model selection for such data sets would prove more problematic. What would be the appropriate asymptotic limit to consider? If we consider a distinguished limit characterised by $N/d \to \alpha < 1$ and $k/N \to \beta < 1$ as $N, d, k \to \infty$, then any asymptotic analysis will need to take account of the effect a non-vanishing proportion of signal population eigenvalues has upon the distribution of sample covariance eigenvalues. The signal directions would no longer represent a small number of rank one perturbations of the identity matrix, with the consequence that the limiting sample covariance eigenvalue distribution would no longer correspond to the Marčenko-Pastur distribution given in Equation (2). Whilst tools exist to characterise the expected sample covariance eigenspectrum for an arbitrary population covariance eigenspectrum (Marčenko and Pastur, 1967; Wachter, 1978; Hoyle and Rattray, 2004b), obtaining closed form analytical results and proving the asymptotic correctness of the model selection for an arbitrary expected sample eigenspectrum would be difficult.

Finally, we should comment that we have illustrated ideas and concepts using model selection for PCA, in particular for $\alpha < 1$. Even today, with readily available compute power and sophisticated statistical learning algorithms, PCA is still a popular tool for dimensionality reduction or exploratory analysis. The application of PCA to extremely high-dimensional small sample size data sets has only increased the need for accurate model selection procedures. We also chose PCA as our exemplar because there already exists an elegant formulation of the Bayesian model selection problem (Minka, 2000, 2001a), and an approximation to the model evidence obtained by routine application of the Laplace approximation had already been developed. However, we believe that many of the ideas presented here are valid more generally. A large contribution to the Bayesian evidence for PCA arises from the rotational degeneracy of the model likelihood, that is, that there are many

orientations of the k-frame formed by the model signal vectors, $\{\boldsymbol{u}_i\}_{i=1}^k$, that are equally capable of accounting for the observed data. This ultimately stems from the fact that we are attempting to make inferences about vectors in $\mathbb{R}^d$ whilst we only have N sample vectors from which to construct a basis for the space. Thus, the degeneracy of the model likelihood is due to a combination of small sample size, $N < d$, and that the likelihood is expressed in terms of projections of the sample data onto the model signal vectors. This is true irrespective of whether the signal vectors $\{\boldsymbol{u}_i\}_{i=1}^k$ are constrained to be orthogonal or not, and so we expect that the issues illustrated here with PCA will be equally applicable to a number of other dimensionality reduction algorithms.

Acknowledgments

The author would like to thank Dr. Magnus Rattray for beneficial discussions and comments on the manuscript.

Appendix A.

Evaluation over data sets of the expectation value, $\mathrm{E}_{\boldsymbol{\xi}}(\sum_{j>k}\ln(\lambda_i-\lambda_j))$ (for $i \leq k$), would appear to be problematic. Since we are interested in the leading order behaviour of this expectation value, that is, the scaling with N, we can change the summation over j to include only those eigenvalues in the bulk distribution given in Equation (2). Potentially the leading order term can then be evaluated via,

$$\lim_{N,d\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right) = \alpha^{-1}\int_{\lambda_{min}}^{\lambda_{max}} d\lambda \ln(\lambda_i-\lambda)\rho_{bulk}(\lambda)\,, \tag{25}$$

and where $\rho_{bulk}(\lambda)$ is the Marčenko-Pastur bulk distribution given in Equation (2). Even if some sample covariance eigenvalues λ_j lie outside the Marčenko-Pastur bulk for $j > k$ the asymptotic result given in (25) is still valid since $N^{-1}\ln(\lambda_i-\lambda) \sim O(N^{-1})$ for $\lambda_i > \lambda > \lambda_{max}$. Thus contributions to $N^{-1}\mathrm{E}_{\boldsymbol{\xi}}(\sum_{j>k}\ln(\lambda_i-\lambda_j))$ from a small number of sample eigenvalues outside of the bulk distribution are vanishingly small in the asymptotic limit. The direct evaluation of the integral in (25) is difficult, so we prefer to use an indirect method. Since we are restricting the summation over j to eigenvalues in the bulk then if we denote the interval $[\lambda_{min},\lambda_{max}] \equiv I_{bulk}$, we can write,

$$\lim_{N,d\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{\lambda_j\in I_{bulk}}\ln(\lambda_i-\lambda_j)\right) = \lim_{N,d\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\mathrm{tr}\ln(\lambda_i\boldsymbol{I} - N^{-1}\boldsymbol{G})\right)\,. \tag{26}$$

where $\boldsymbol{G}$ is the Gram matrix formed from N, d-dimensional samples drawn from a multi-variate zero-mean Gaussian distribution with population covariance $\boldsymbol{C} = \sigma^2\boldsymbol{I}$, that is, the matrix $\boldsymbol{G}$ has elements $G_{\mu\mu'} = \boldsymbol{\xi}_\mu^T\boldsymbol{\xi}_{\mu'}$. The expectation, over data sets $\{\boldsymbol{\xi}_\mu\}_{\mu=1}^N$, of $\mathrm{tr}\ln(\lambda_i\boldsymbol{I} - N^{-1}\boldsymbol{G})$ is performed with the aid of the replica trick, which uses the representation,

$$\ln y = \lim_{n\to 0}\frac{(y^n-1)}{n}\,.$$

The calculation proceeds in a straight-forward fashion. We only give brief details here and the reader is referred to more in-depth explanations, given elsewhere, of the use of the replica trick in statistical physics and machine learning (Mezard et al., 1987; Hertz et al., 1991; Engel and Van den

Broeck, 2001). We find that evaluation of Equation (26) is given by the extremal value (with respect to x and q_1) of,

$$-\left\{\ln x + \frac{q_1}{x} - \alpha^{-1}\ln(1-\sigma^2 x) - \frac{\alpha^{-1}\sigma^2 q_1}{1-\sigma^2 x} - \lambda_i(x+q_1) + 1\right\}. \tag{27}$$

Differentiating with respect to x and q_1 the expression in (27) is easily maximized to give (for $\lambda_i = \sigma^2(1+A_i)(1+\alpha^{-1}A_i^{-1})$),

$$\ln\sigma^2 - (\alpha^{-1}-1)\ln(1+A_i) + \alpha^{-1}\ln A_i + \frac{1}{\alpha A_i}. \tag{28}$$

Here we have assumed the sample covariance eigenvalue λ_i will correspond to that from a "spiked" population covariance and that we are above the retarded learning transition for the i^{th} signal component. For $\alpha < A_i^{-2}$ we are below the retarded learning transition for the i^{th} signal and we expect λ_i to be located approximately at the upper edge of the bulk distribution so that $\lambda_i \simeq \sigma^2(1+\alpha^{-\frac{1}{2}})^2$. We expect the summation in $N^{-1}\sum_{j>k}\ln(\lambda_i-\lambda_j)$ will still converge since it is restricted to $j > k \geq i$. Setting $\lambda_i = \lambda_{max}$ in the previous replica calculation still yields a well-behaved estimate for $\lim_{N\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right)$, namely,

$$\ln\sigma^2 - (\alpha^{-1}-1)\ln(1+\alpha^{-\frac{1}{2}}) + \alpha^{-1}\ln\alpha^{-\frac{1}{2}} + \alpha^{-\frac{1}{2}}. \tag{29}$$

Since $A_i = \alpha^{-\frac{1}{2}}$ is the limit at which λ_i is indistinguishable from the bulk distribution, that is, $\lambda_i \to \lambda_{max}$, it is unsurprising that Equation (29) is obtained as the limit of Equation (28) as $A_i \to \alpha^{-\frac{1}{2}}$.

Figure 6a compares the limiting theoretical estimates for $N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right)$, given in Equations (28) and (29) with simulation for different values of α. Different plotted symbols represent different values of i, with $i = 1,\ldots,5$ running from top to bottom respectively. For each series we have set $k = i$ in the evaluation of the simulation averages. This was considered to be better than artificially setting $k = 3$, the true signal dimensionality which in general would not be known. Although in some cases, for evaluation of the simulation averages, this will lead to summation over sample covariance eigenvalues that are outside of the Marčenko-Pastur bulk distribution these will make only $O(N^{-1})$ contributions to $N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right)$, and so the simulation averages still provide a relevant test of the theoretical estimate of the asymptotic limiting value $\lim_{N\to\infty} N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right)$. Asymptotically, in the limit $N\to\infty$ and for the population covariance signal strengths chosen, three sample covariance eigenvalues are expected to be separated from the bulk distribution over the entire range of α plotted. Consequently we expect simulation averages to have a distinctly different behaviour for $i \leq 3$ compared to $i > 3$. A common limiting value for $N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>k}\ln(\lambda_i-\lambda_j)\right)$ when $i > 3$ is apparent from Figure 6a. Figure 6b compares simulation averages with the theoretical estimates in Equations (28) and (29) for different signal strengths. In this case the population covariance contains a single signal component, of strength A, whilst we have fixed $\alpha = 0.1$. The sample covariance eigenspectrum is expected to display a transition at $A = 1.0/\sqrt{\alpha} \simeq 3.16$. This is clearly reflected in the behaviour of the simulation average. The convergence towards the limiting theoretical estimate is also apparent from the comparison of simulation averages for $d = 1000$ and $d = 2000$.

References

T.W. Anderson. Asymptotic theory for principal component analysis. *Annals of Mathematical Statistics*, 34:122–148, 1963.

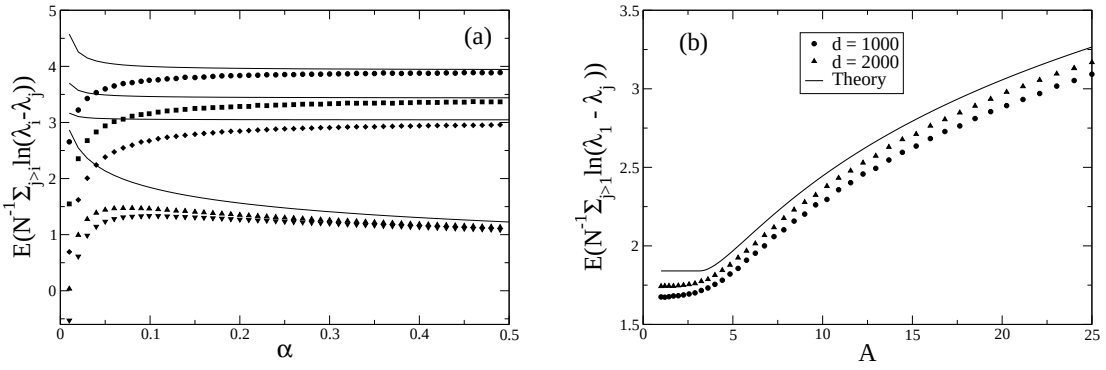


Figure 6: Comparison of simulation averages of $N^{-1}\mathrm{E}_{\boldsymbol{\xi}}\left(\sum_{j>i}\ln(\lambda_i-\lambda_j)\right)$ with the limiting theoretical estimates given in Equations (28) and (29). Figure a shows behaviour of the expectation value with α for $i = 1,\ldots,5$. Solid symbols show simulation averages whilst the solid lines show the corresponding theoretical estimates. We have set $d = 1000$ and $\sigma^2 = 1$. The population covariance contains three signal components with $A_1 = 50, A_2 = 30, A_3 = 20$. Figure b shows comparison of the theoretical result with simulation for different signal strengths, at two different values of the data dimensionality d. We have set $\alpha = 0.1, \sigma^2 = 1$. The population covariance contains a single signal component with signal strength A. For both Figure a and Figure b simulation averages are taken over 1000 matrices, and error bars of the simulation averages are smaller than the size of the plotted symbols.

J. Baik and J.W. Silverstein. Eigenvalues of large sample covariance matrices of spiked population models. *Journal of Multivariate Analysis*, 97:1382–1408, 2006.

J. Baik, G. Ben Arous, and S. Peche. Phase transition of the largest eigenvalue for non-null complex sample covariance matrices. *Annals of Probability*, 33:1643–1697, 2005.

C.M. Bishop. Bayesian PCA. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Advances in Neural Information Processing Systems*, pages 382–388. MIT Press, 1999a.

C.M. Bishop. Variational principal components. In *Proceedings Ninth International Conference on Artificial Neural Networks, ICANN'99*, pages 509–514. IEE, 1999b.

A. Engel and C. Van den Broeck. *Statistical Mechanics of Learning*. CUP, Cambridge, 2001.

T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286: 531–537, 1999.

J.A. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation (Santa Fe Institute Studies in the Sciences of Complexity)*. Addison-Wesley, Redwood City, CA, 1991.

D.C. Hoyle and M. Rattray. PCA learning for sparse high-dimensional data. *Europhysics Letters*, 62:117–123, 2003.

D.C. Hoyle and M. Rattray. Principal-component-analysis eigenvalue spectra from data with symmetry-breaking structure. *Physical Review E*, 69:026124, 2004a.

D.C. Hoyle and M. Rattray. Statistical mechanics of learning multiple orthogonal signals : asymptotic theory and fluctuation effects. *Physical Review E*, 75:016101, 2007.

D.C. Hoyle and M. Rattray. A statistical mechanics analysis of gram matrix eigenvalue spectra. In J. Shawe-Taylor and Y. Singer, editors, *Proceedings of COLT'04, Conference on Learning Theory, Banff, Canada, 2004. Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2004b.

A.T. James. Normal multivariate analysis and the orthogonal group. *Annals of Mathematical Statistics*, 25:40–75, 1954.

S. John. Some optimal multivariate tests. *Biometrika*, 58:123–127, 1971.

I.M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of Statistics*, 29:295–327, 2001.

I.M. Johnstone. High dimensional statistical inference and random matrices. In M. Sanz-Solé, J. Soria, J.L. Varona, and J. Verdera, editors, *Proceedings of International Congress of Mathematicians, Madrid, 2006*. European Mathematical Society Publishing House, 2006.

I.T. Joliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

D. Landgrebe. Hyperspectral image data analysis as a high dimensional signal processing problem. *IEEE Signal Processing Magazine*, 19:17–28, 2002.

O. Ledoit and M. Wolf. Some hypothesis tests for the covariance matrix when the dimension is large compared to the sample size. *Annals of Statistics*, 30:1081–1102, 2002.

D.J.C. MacKay. Choice of basis for Laplace approximation. *Machine Learning*, 33:77–86, 1998.

D.J.C MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

V.A. Marčenko and L.A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Math. USSR-Sb*, 1:457–483, 1967.

M. Mezard, G. Parisi, and M. Virasoro. *Spin Glass Theory and Beyond*. World Scientific Publishing, Singapore, 1987.

T.P. Minka. Automatic choice of dimensionality for PCA. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *NIPS 13*, pages 598–604. MIT Press, 2001a.

T.P. Minka. Automatic choice of dimensionality for PCA. Technical Report TR-514, M.I.T. Media Laboratory Perceptual Computing Section, 2000. Available from http://vismod.media.mit.edu/tech-reports/TR-514-ABSTRACT.html.

T.P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, UAI-2001*, pages 362–369, 2001b.

H. Nagao. On some test criteria for covariance matrix. *Annals of Statistics*, 1:700–709, 1973.

A.L. Price, N.J. Patterson, R.M. Plenge, M.A. Weinblatt, N.A. Shadick, and D. Reich. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38:904–909, 2006.

P. Reimann, C. Van den Broeck, and G.J. Bex. A gaussian scenario for unsupervised learning. *Journal of Physics A:Mathematical and General.*, 29:3521–3535, 1996.

S. Roweis. EM algorithms for PCA and SPCA. In M I. Jordan, M J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1998.

B. Scholköpf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

M.E. Tipping and C.M. Bishop. Probabilistic principal component analysis. *J. Royal Statistical Society B*, 61:611–622, 1999a.

M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11:443–482, 1999b.

C.A. Tracy and H. Widom. On orthogonal and symplectic matrix ensembles. *Communications in Mathematical Physics*, 177:727–754, 1996.

R. Urbanczik. Statistical physics of independent component analysis. *Europhysics Letters*, 64: 564–570, 2003.

V. Šmídl and A. Quinn. On Bayesian principal component analysis. *Computational Statistics and Data Analysis*, 51:4101–4123, 2007.

K.W. Wachter. In David C. Hoaglin & Roy E. Welsch, editor, *Proceedings of the Ninth Interface Symposium Computer Science and Statistics*, page 299, Boston, 1976. Prindle, Weber and Schmidt.

K.W. Wachter. The strong limits of random matrix spectra for sample matrices of independent elements. *Annnals Probability*, 6:1–18, 1978.

R. Wong. *Asymptotic Approximations of Integrals*. Academic Press, Boston, MA, 1989.

Journal of Machine Learning Research 9 (2008) 2761-2801 Submitted 1/08; Revised 9/08; Published 12/08

Robust Submodular Observation Selection

Andreas Krause KRAUSEA@CS.CMU.EDU
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213

H. Brendan McMahan MCMAHAN@GOOGLE.COM
Google, Inc.
4720 Forbes Ave.
Pittsburgh, PA 15213

Carlos Guestrin GUESTRIN@CS.CMU.EDU
Computer Science Department and Machine Learning Department
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213

Anupam Gupta ANUPAMG@CS.CMU.EDU
Computer Science Department
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213

Editor: Chris Williams

Abstract

In many applications, one has to actively select among a set of expensive observations before making an informed decision. For example, in environmental monitoring, we want to select locations to measure in order to most effectively predict spatial phenomena. Often, we want to select observations which are robust against a number of possible objective functions. Examples include minimizing the maximum posterior variance in Gaussian Process regression, robust experimental design, and sensor placement for outbreak detection. In this paper, we present the *Submodular Saturation* algorithm, a simple and efficient algorithm with strong theoretical approximation guarantees for cases where the possible objective functions exhibit *submodularity*, an intuitive diminishing returns property. Moreover, we prove that better approximation algorithms do not exist unless NP-complete problems admit efficient algorithms. We show how our algorithm can be extended to handle complex cost functions (incorporating non-unit observation cost or communication and path costs). We also show how the algorithm can be used to near-optimally trade off expected-case (e.g., the Mean Square Prediction Error in Gaussian Process regression) and worst-case (e.g., maximum predictive variance) performance. We show that many important machine learning problems fit our robust submodular observation selection formalism, and provide extensive empirical evaluation on several real-world problems. For Gaussian Process regression, our algorithm compares favorably with state-of-the-art heuristics described in the geostatistics literature, while being simpler, faster and providing theoretical guarantees. For robust experimental design, our algorithm performs favorably compared to SDP-based algorithms.

Keywords: observation selection, experimental design, active learning, submodular functions, Gaussian processes

1. Introduction

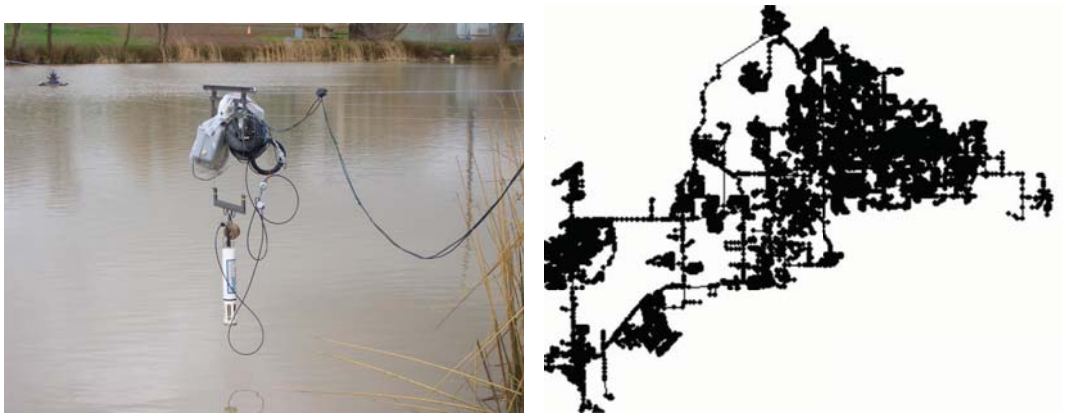
In tasks such as sensor placement for environmental monitoring or experimental design, one has to select among a large set of possible, but expensive, observations. In environmental monitoring, we can choose locations where measurements of a spatial phenomenon (such acidicity in rivers and lakes, cf., Figure 1(a)) should be obtained. In experimental design, we frequently have a menu of possible experiments which can be performed. Often, there are several different objective functions which we want to simultaneously optimize. For example, in the environmental monitoring problem, we want to minimize the marginal posterior variance of our acidicity estimate at all locations simultaneously. In experimental design, we often have uncertainty about the model parameters, and we want our experiments to be informative no matter what the true parameters of the model are. In sensor placement for contamination detection in water distribution networks (cf., Figure 1(b)), we want to place sensors in order to quickly detect any possible contamination event.

Our goal in all these problems is to select observations (sensor locations, experiments) which are *robust* against a worst-case objective function (location to evaluate predictive variance, model parameters, contamination event, etc.). Often, the individual objective functions, for example, the marginal variance at one location, or the information gain for a fixed set of parameters (Das and Kempe, 2008; Krause et al., 2007b; Krause and Guestrin, 2005; Guestrin et al., 2005), satisfy *submodularity*, an intuitive diminishing returns property: Adding a new observation helps less if we have already made many observations, and more if we have made few observation thus far. While NP-hard, the problem of selecting an optimal set of k observations maximizing a single submodular objective can be approximately solved using a simple greedy forward-selection algorithm, which is guaranteed to perform near-optimally (Nemhauser et al., 1978). However, as we show, this simple *myopic* algorithm performs arbitrarily badly in the case of a worst-case objective function. In this paper, we address the fundamental problem of nonmyopically selecting observations which are robust against such an adversarially chosen submodular objective function. In particular:

- We present SATURATE, an efficient algorithm for the robust submodular observation selection problem. Our algorithm guarantees solutions which are at least as informative as the optimal solution, at only a slightly higher cost.
- We prove that our approximation guarantee is the best possible, that is, the guarantee cannot be improved unless NP-complete problems admit efficient algorithms.
- We discuss several extensions of our approach, handling complex cost functions and trading off worst-case and average-case performance.
- We extensively evaluate our algorithm on several real-world tasks, including minimizing the maximum posterior variance in Gaussian Process regression, finding experiment designs which are robust with respect to parameter uncertainty, and sensor placement for outbreak detection.

This manuscript is organized as follows. In Section 2, we formulate the robust submodular observation selection problem, and in Section 3, we analyze its hardness. We subsequently present

SATURATE, an efficient approximation algorithm for this problem (Section 4), and show that our approximation guarantees are best possible, unless NP-complete problems admit efficient algorithms (Section 5). In Section 6, we discuss how many important machine learning problems are instances of our robust submodular observation selection formalism. We then discuss extensions (Section 7) and evaluate the performance of SATURATE on several real-world observation selection problems (Section 8). Section 9 presents heuristics to improve the computational performance of our algorithm, Section 10 reviews related work, and Section 11 presents our conclusions.



(a) *NIMS deployed at UC Merced* (b) *Water distribution network*

Figure 1: (a) Deployment of the Networked Infomechanical System (NIMS, Harmon et al., 2006) to monitor a lake near UC Merced. (b) Illustration of the municipal water distribution network considered in the Battle of the Water Sensor Networks challenge (cf., Ostfeld et al., 2008).

2. Robust Submodular Observation Selection

In this section, we first review the concept of submodularity (Section 2.1), and then introduce the *robust submodular observation selection (*RSOS*)* problem (Section 2.2).

2.1 Submodular Observation Selection

Let us consider a spatial prediction problem, where we want to estimate the pH values across a horizontal transect of a river, for example, using the NIMS robot shown in Figure 1(a). We can discretize the space into a finite number of locations V, where we can obtain measurements, and model a joint distribution $P(X_V)$ over variables X_V associated with these locations. One example of such models, which have found common use in geostatistics (cf., Cressie, 1991), are Gaussian Processes (cf., Rasmussen and Williams, 2006). Based on such a model, a typical goal in spatial monitoring is to select a subset of locations $A \subseteq V$ to observe, such that the average predictive variance,

$$V(A) = \frac{1}{n}\sum_i \sigma^2_{i|A},$$

is minimized (cf., Section 6.1 for more details). Hereby, $\sigma^2_{i|A}$ denotes the predictive variance at location i after observing locations A, that is,

$$\sigma^2_{i|A} = \int P(\mathbf{x}_A)\mathbb{E}\left[(X_i - \mathbb{E}[X_i \mid \mathbf{x}_A])^2 \mid \mathbf{x}_A\right] d\mathbf{x}_A.$$

Unfortunately, the problem

$$A^* = \operatorname*{argmin}_{|A|\leq k} V(A)$$

is NP-hard in general (Das and Kempe, 2008), and the number of candidate solutions is very large, so generally we cannot expect to efficiently find the optimal solution. Fortunately, as Das and Kempe (2008) show, in many cases, the *variance reduction*

$$F_s(A) = \sigma^2_s - \sigma^2_{s|A}$$

at any particular location s, satisfies the following diminishing returns behavior: Adding a new observation reduces the variance at s more, if we have made few observations so far, and less, if we have already made many observations. This formalism can be formalized using the combinatorial concept of *submodularity* (cf., Nemhauser et al., 1978):

Definition 1 *A set function $F : 2^V \to \mathbb{R}$ is called* submodular, *if for all subsets $A, B \subseteq V$ it holds that $F(A \cup B) + F(A \cap B) \leq F(A) + F(B)$.*

Nemhauser et al. (1978) prove a convenient characterization of submodular functions: F is submodular if and only if for all $A \subseteq B \subseteq V$ and $s \in V \setminus B$ it holds that $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$. This characterization exactly matches our diminishing returns intuition about the variance reduction F_s at location s. Since each of the variance reduction functions F_s is submodular, the *average variance reduction*

$$F(A) = V(\emptyset) - V(A) = \frac{1}{n}\sum_s F_s(A)$$

is also submodular. The average variance reduction is also *monotonic*, that is, for all $A \subseteq B \subseteq V$ it holds that $F(A) \leq F(B)$, and *normalized* ($F(\emptyset) = 0$).

Hence, the problem of minimizing the average variance is an instance of the problem

$$\max_{A\subseteq V} F(A), \quad \text{subject to} \quad |A| \leq k, \tag{1}$$

where F is normalized, monotonic and submodular, and k is a bound on the number of observations we can make. As Krause and Guestrin (2007a) show, many other observation selection problems are instances of Problem (1).

Since solving Problem (1) is NP-hard in most interesting instances (Feige, 1998; Krause et al., 2006, 2007b; Das and Kempe, 2008), in practice, heuristics are often used. One such heuristic is the *greedy algorithm*. This algorithm starts with the empty set, and iteratively adds the element $s^* = \operatorname{argmax}_{s\in V\setminus A} F(A \cup \{s\})$, until k elements have been selected. Perhaps surprisingly, a fundamental result by Nemhauser et al. (1978) states that for submodular functions, the greedy algorithm achieves a constant factor approximation:

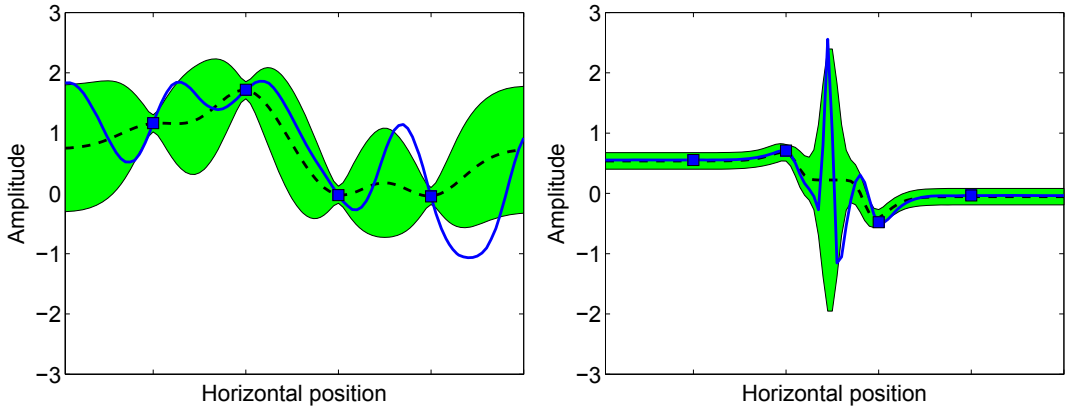
Theorem 2 (Nemhauser et al. 1978) *In the case of any normalized, monotonic submodular function F, the set A_G obtained by the greedy algorithm achieves at least a constant fraction $(1-1/e)$ of the objective value obtained by the optimal solution, that is,*

$$F(A_G) \geq (1-1/e) \max_{|A|\leq k} F(A).$$

Moreover, no polynomial time algorithm can provide a better approximation guarantee unless P = NP (Feige, 1998).

2.2 The Robust Submodular Observation Selection (RSOS) Problem

For phenomena, such as the one indicated in Figure 2(a), which are spatially homogeneous (isotropic), maximizing this average variance reduction leads to effective variance reduction everywhere in the space. However, many spatial phenomena are nonstationary, being smooth in certain areas and highly variable in others, such as the example indicated in Figure 2(b). In such a case, maximizing the average variance reduction will typically put only few examples in the areas highly variable areas. However, those regions are typically the most interesting, since they are most difficult to predict. In such cases, we might want to simultaneously minimize the variance everywhere in the space.



(a) *High average variance* (b) *High maximum variance*

Figure 2: Spatial predictions using Gaussian Processes with a small number of observations. The blue solid line indicates the unobserved latent function, and blue squares indicate observations. The plots also show confidence bands (green). Dashed line indicates the prediction. (b) shows an example with high maximum predictive variance, but low average variance, whereas (a) shows an example with high average variance, but lower maximum variance. Note, that in (b) we are most uncertain about the most variable (and interesting, since it is hard to predict) part of the curve, suggesting that the maximum variance should be optimized.

More generally, in many applications (such as the spatial monitoring problem discussed above, and several other examples which we present in Section 6), we want to perform equally well with respect to *multiple* objectives. We will hence consider settings where we are given a *collection* of

A	$F_1(A)$	$F_2(A)$	$\min_i F_i(A)$
$\emptyset$	0	0	0
$\{s_1\}$	n	0	0
$\{s_2\}$	0	n	0
$\{t_1\}$	1	1	1
$\{t_2\}$	1	1	1
$\{s_1,s_2\}$	n	n	n
$\{s_1,t_1\}$	$n+1$	1	1
$\{s_1,t_2\}$	$n+1$	1	1
$\{s_2,t_1\}$	1	$n+1$	1
$\{s_2,t_2\}$	1	$n+1$	1
$\{t_1,t_2\}$	2	2	2

Table 1: Functions F_1 and F_2 used in the counterexample.

normalized monotonic submodular functions $F_1,\ldots,F_m$, and we want to solve

$$\max_{A\subseteq V} \min_i F_i(A), \quad \text{subject to} \quad |A| \leq k. \tag{2}$$

The goal of Problem (2) is to find a set A of observations, which is robust against the worst possible objective, $\min_i F_i$, from our set of possible objectives. Consider the spatial monitoring setting for example, and assume that the prior variance σ_i^2 is constant (we will relax this assumption in Section 7.2) over all locations i. Then, the problem of minimizing the maximum variance, as motivated by the example in Figure 2, is equivalent to maximizing the minimum variance reduction, that is, solving Problem (2) where F_i is the variance reduction at location i.

We call Problem (2) the *Robust Submodular Observation Selection (*RSOS*)* problem. Note, that even if the F_i are all submodular, $F_{wc}(A) = \min_i F_i(A)$ is generally *not* submodular. In fact, we show below that, in this setting, the simple greedy algorithm (which performs near-optimally in the single-criterion setting) can perform arbitrarily badly. While the example in Table 1 might seem artificial, as we show in Section 8 (especially Section 8.3), the greedy algorithm exhibits very poor performance when applied to practical problems.

3. Hardness of the Robust Submodular Observation Selection Problem

Given the near-optimal performance of the greedy algorithm for the single-objective problem, a natural question is if the performance guarantee generalizes to the more complex robust optimization setting. Unfortunately, this hope is far from true, even in the simpler case of *modular* (additive) functions F_i. Consider a case with two submodular functions, F_1 and F_2, where the set of observations is $V = \{s_1,s_2,t_1,t_2\}$. The functions take values as indicated in Table 1. Optimizing for a set of 2 elements, the greedy algorithm maximizing $F_{wc}(A) = \min\{F_1(A),F_2(A)\}$ would first choose t_1 (or t_2), as this choice increases the objective $\min\{F_1,F_2\}$ by 1, as opposed to 0 for s_1 and s_2. The greedy solution for $k = 2$ would then be the set $\{t_1,t_2\}$, obtaining a score of 2. However, the optimal solution with $k = 2$ is $\{s_1,s_2\}$, with a score of n. Hence, as $n \to \infty$, the greedy algorithm performs arbitrarily worse than the optimal solution.

Given that the greedy algorithm performs arbitrarily badly, our next hope would be to obtain a different good approximation algorithm. However, we can show that most likely this is not possible:

Theorem 3 *Unless* P = NP, *there cannot exist* any *polynomial time approximation algorithm for Problem* (2). *More precisely: If there exists a positive function* $\gamma(\cdot) > 0$ *and an algorithm that, for all n and k, in time polynomial in the size of the problem instance n, is guaranteed to find a set* A' *of size k such that* $\min_i F_i(A') \geq \gamma(n) \max_{|A|\leq k} \min_i F_i(A)$, *then* P = NP.

Thus, unless P = NP, there cannot exist any algorithm which is guaranteed to provide, for example, even an exponentially small fraction ($\gamma(n) = 2^{-n}$) of the optimal solution. All proofs can be found in the Appendix.

4. The Submodular Saturation Algorithm

We now present an algorithm that finds a set of observations which perform at least as well as the optimal set, but at slightly increased cost; moreover, we show that no efficient algorithm can provide better guarantees (under reasonable complexity-theoretic assumptions).

4.1 Algorithm Overview

For now we assume that all F_i take only integral values; this assumption is relaxed in Section 7.1. The key idea is to consider the following alternative problem formulation:

$$\max_{c,A} c, \qquad \text{subject to} \qquad F_i(A) \geq c \text{ for } 1 \leq i \leq m \text{ and } |A| \leq k. \tag{3}$$

We want a set A of size at most k, such that $F_i(A) \geq c$ for all i, and c is as large as possible. Note that Problem (3) is equivalent to the original Problem (2): Maximizing c subject to the existence of a set A, $|A| \leq k$ such that $F_i(A) \geq c$ for all i is equivalent to maximizing $\min_i F_i(A)$.

Now suppose we had an algorithm that, for any given value c, solves the following optimization problem:

$$A_c = \operatorname*{argmin}_A |A| \qquad \text{subject to} \qquad F_i(A) \geq c \text{ for } 1 \leq i \leq m \tag{4}$$

that is, finds the smallest set A with $F_i(A) \geq c$ for all i. If this set has at most k elements, then c (and the set A) is feasible for the RSOS Problem (3). If we cannot find a set A satisfying $F_i(A) \geq c$ for all i and containing at most k elements, then c is infeasible. A binary search on c would then allow us to find the optimal solution with the maximum feasible c. We call Problem (4) the MINCOVER$_c$ problem, as it requires to find the smallest set guaranteeing an equal amount of coverage, c, for all objective functions F_i.

Since Theorem 3 rules out *any* approximation algorithm which respects the constraint k on the size of the set A, our only hope for non-trivial guarantees requires us to relax this constraint. Our algorithm is based on the following approach:

- We define a relaxed version of the RSOS problem with a superset of feasible solutions that we call RELRSOS.
- We will maintain a lower bound (a feasible solution) for RELRSOS, and an upper bound for RSOS.

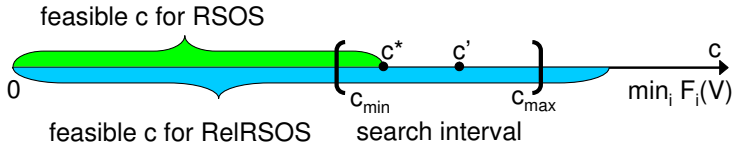


Figure 3: Illustration of feasible regions for the RSOS and RELRSOS problems. $[c_{\min}, c_{\max}]$ is the search interval during some iteration of SATURATE. c^* is the optimal solution to the RSOS problem, and c' is the solution that will eventually be returned by SATURATE.

- We will then successively improve the upper and lower bounds using a binary search procedure. Upon convergence, we are thus guaranteed a feasible solution to RELRSOS, that performs at least as well as the optimal solution to the RSOS problem.

We now define the RELRSOS problem, the relaxed version of the RSOS Problem (3).

$$\max_{c,A} c, \qquad \text{subject to} \quad F_i(A) \geq c \text{ for } 1 \leq i \leq m \text{ and } |A| \leq \alpha k. \tag{5}$$

Hereby, $\alpha \geq 1$ is a parameter relaxing the constraint on $|A|$. If $\alpha = 1$, we recover the RSOS Problem (3).

As described above, our goal will be to *approximately* solve the RELRSOS Problem (5) for a fixed constant α. More formally, we will develop an efficient algorithm, SATURATE, which returns a solution (c', A') that is feasible for the RELRSOS Problem (5), and achieves a score that is at least as good as an optimal solution (c^*, A^*) to the RSOS Problem (3), that is, $c' \geq c^*$ and $|A'| \leq \alpha |A^*| \leq \alpha k$.

The basic idea of SATURATE is to use the binary search procedure (maintaining a search interval $[c_{\min}, c_{\max}]$) as described above, but using an *approximate* algorithm, GPC (for *Greedy Partial Coverage*) that we will develop below, for the MINCOVER$_c$ Problem (4). When invoked with a fixed value c, the GPC algorithm will return a feasible solution $|A'_c|$ to the MINCOVER$_c$ Problem (4). We will furthermore guarantee that

- $|A'_c| > \alpha k$ implies that $c > c^*$, that is, c is an upper bound to the RSOS Problem (3), and hence it is safe to set $c_{\max} = c$, and
- $|A'_c| \leq \alpha k$ implies that A'_c is a feasible solution (lower bound) to the RELRSOS Problem (5). A'_c is then kept as best current solution and we can set $c_{\min} = c$.

The binary search procedure will hence always maintain an upper bound $c_{\max}$ to the RSOS Problem (3), and a lower bound $c_{\min}$ to the RELRSOS Problem (5). Upon termination, it is thus guaranteed to find a solution A_S for which it holds that $\min_i F_i(A_S) \geq c^*$ (since A_S is an upper bound to the RSOS Problem (3)) and $|A_S| \leq \alpha k$ (since A_S is feasible for the RELRSOS Problem (5)). Hence, the approximate solution A_S obtains minimum value at least as high as the best possible score obtainable using k elements, but using slightly more (at most αk) elements than k elements. Figure 3 illustrates the feasible regions of the RSOS and RELRSOS problems, as well as the binary search procedure.

4.2 Algorithm Details

We will now provide formal details for the algorithm sketched in Section 4.1. As trivial lower and upper bounds for the RSOS problem we can initially set $c_{\min} = 0 \leq \min_i F_i(\emptyset)$, and $c_{\max} = \min_i F_i(V)$, due to monotonicity of the F_i.

First, we will develop the efficient algorithm GPC which approximately solves the MINCOVER$_c$ Problem (4). For any value c that could possibly be feasible (i.e., $0 \leq c \leq \min_i F_i(V)$), define $\widehat{F}_{i,c}(A) = \min\{F_i(A), c\}$, the original function F_i truncated at score level c. The key insight is that these truncated functions $\widehat{F}_{i,c}$ remain monotonic and submodular (Fujito, 2000). Figure 4 illustrates this truncation concept. Let $\overline{F}_c(A) = \frac{1}{m}\sum_i \widehat{F}_{i,c}(A)$ be their average value. Since monotonic submodular functions are closed under convex combinations, $\overline{F}_c$ is also submodular and monotonic. Furthermore, $F_i(A) \geq c$ for all $1 \leq i \leq m$ if and only if $\overline{F}_c(A) = c$. Hence, in order to determine whether some c is feasible for Problem (5), we need to determine whether there exists a set of size at most αk such that $\overline{F}_c(A) = c$. Note, that due to monotonicity of $\overline{F}_c$ and the choice of c it holds that that $c = \overline{F}_c(V)$. We hence need to solve the following optimization problem:

$$A_c^* = \operatorname*{argmin}_{A \subseteq V} |A|, \quad \text{such that} \quad \overline{F}_c(A) = \overline{F}_c(V). \tag{6}$$

Problems of the form $\min_A |A|$ such that $F(A) = F(V)$, where F is a monotonic submodular function, are called *submodular covering problems*. Since $\overline{F}_c$ satisfies these requirements, the MINCOVER$_c$ Problem (6) is an instance of such a submodular covering problem. While such problems are NP-hard in general (Feige, 1998), Wolsey (1982) shows that the greedy algorithm, that starts with the empty set ($A = \emptyset$) and iteratively adds the element s increasing the score the most until $F(A) = F(V)$, achieves near-optimal performance on this problem. We can hence use the greedy algorithm applied to the truncated functions $\overline{F}_c$ as our approximate algorithm *GPC*, which is formalized in Algorithm 1. Using Wolsey's result and the observation that α can be chosen independently of the truncation threshold c, we find:

Lemma 4 *Given integral valued*[1] *monotonic submodular functions $F_1, \ldots, F_m$ and a (feasible) constant c, Algorithm 1 (with input $\overline{F}_c$) finds a set A_G such that $F_i(A_G) \geq c$ for all i, and $|A_G| \leq \alpha |A_c^*|$, where A_c^* is an optimal solution to Problem* (6)*, and*

$$\alpha = 1 + \log\left(\max_{s \in V} \sum_i F_i(\{s\})\right).$$

We can compute this approximation guarantee α for any given instance of the RSOS problem. Hence, if for a given value of c the greedy algorithm returns a set of size greater than αk, there cannot exist a solution A' with $|A'| \leq k$ with $F_i(A') \geq c$ for all i. Thus, c is an upper bound to the RSOS Problem (3). We can use this argument to conduct the binary search discussed in Section 4.1 to find the optimal value of c. The binary search procedure maintains an interval $[c_{\min}, c_{\max}]$, initialized $[0, \min_i F_i(V)]$. At every iteration, we test the current center of the interval, $c = (c_{\min} + c_{\max})/2$, and check feasibility of c using the greedy algorithm. If c is feasible, we retain the current best feasible solution and set $c_{\min} = c$. If c is infeasible (which we detect by comparing the number of elements picked by the greedy algorithm with αk), we set $c_{\max} = c$.

1. This bound is only meaningful for integral F_i, otherwise it could be arbitrarily improved by scaling the F_i. We relax the constraint on integrality of the F_i in Section 7.1.

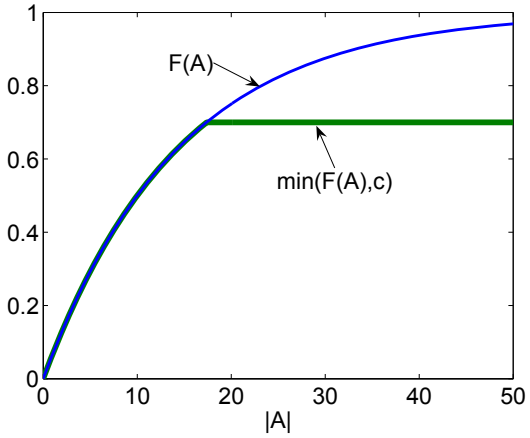


Figure 4: Truncating an objective function F preserves submodularity and monotonicity.

GPC ($\overline{F}_c$, c)
$A \leftarrow \emptyset$;
while $\overline{F}_c(A) < c$ **do**
  foreach $s \in V \setminus A$ **do** $\delta_s \leftarrow \overline{F}_c(A \cup \{s\}) - \overline{F}_c(A)$;
  $A \leftarrow A \cup \{\text{argmax}_s \, \delta_s\}$;
end

Algorithm 1: The greedy submodular partial cover (GPC) algorithm.

We call Algorithm 2, which formalizes this procedure, the *submodular saturation algorithm* (SATURATE), as the algorithm considers the truncated objectives $\widehat{F}_{i,c}$, and chooses sets which *saturate* all these objectives. In the pseudo-code of Algorithm 2 we pass α as a parameter. Theorem 5 (given below) states that SATURATE, when applied with α chosen as in Lemma 4, is guaranteed to find a set which achieves worst-case score $\min_i F_i$ at least as high as the optimal solution, if we allow the set to be logarithmically (a factor α) larger than the optimal solution.

Theorem 5 *For any integer k,* SATURATE *finds a solution A_S such that*

$$\min_i F_i(A_S) \geq \max_{|A| \leq k} \min_i F_i(A) \quad \text{and} \quad |A_S| \leq \alpha k,$$

for $\alpha = 1 + \log\left(\max_{s \in V} \sum_i F_i(\{s\})\right)$. The total number of submodular function evaluations is

$$O\left(|V|^2 m \log\left(m \min_i F_i(V)\right)\right).$$

Note, that the algorithm still makes sense for any value of α. However, if $\alpha < 1 + \log\left(\max_{s \in V} \sum_i F_i(\{s\})\right)$, the guarantee of Theorem 5 does not hold. As argued in Section 4.1, if we had an exact algorithm for submodular coverage, then we would set $\alpha = 1$, and SATURATE would return the optimal solution to the RSOS problem. Since, in our experience, the greedy algorithm for optimizing submodular functions works very effectively (cf., Krause et al. 2007b), in our experiments, we call SATURATE with $\alpha = 1$. This choice empirically performs very well, as demonstrated in Section 8.

SATURATE $(F_1,\ldots,F_m,k,\alpha)$
$c_{\min} \leftarrow 0$; $c_{\max} \leftarrow \min_i F_i(V)$; $A_{best} \leftarrow \emptyset$;
while $(c_{\max} - c_{\min}) \geq \frac{1}{m}$ **do**
  $c \leftarrow (c_{\min} + c_{\max})/2$;
  Define $\overline{F}_c(A) \leftarrow \frac{1}{m}\sum_i \min\{F_i(A), c\}$;
  $\widehat{A} \leftarrow GPC(\overline{F}_c, c)$;
  if $|\widehat{A}| > \alpha k$ **then**
    $c_{\max} \leftarrow c$;
  else
    $c_{\min} \leftarrow c$; $A_{best} = \widehat{A}$
  end
end

Algorithm 2: The Submodular Saturation algorithm.

If we apply SATURATE to the example problem described in Section 3, we would start with $c_{\max} = n$. Running the coverage algorithm (GPC) with $c = n/2$ would first pick element s_1 (or s_2), since $\overline{F}_c(\{s_1\}) = n/2$, and, next, pick s_2 (or s_1 resp.), hence finding the optimal solution.

The worst-case running time guarantee is quite pessimistic, and in practice the algorithm is much faster: Using a priority queue and lazy evaluations, Algorithm 1 can be sped up drastically. Lazy evaluations exploit the fact that, due to submodularity, the differences $\delta_s(A) = \overline{F}_c(X_{A\cup s}) - \overline{F}_c(X_A)$ that are computed by GPC are monotonically decreasing in A, which allows to avoid a large number of function evaluations (cf., Robertazzi and Schwartz 1989 for details). In addition, for many submodular functions F_i, such as the variance reduction, it is often cheaper to compute $\overline{F}_c(X_{A\cup s}) - \overline{F}_c(X_A)$ instead of $\overline{F}_c(X_{A\cup s})$. This observation can be exploited to drastically speed up GPC. Furthermore, in practical implementations, one would stop GPC once $\alpha k + 1$ elements have been selected, which already proves that the optimal solution with k elements cannot achieve score c. Also, Algorithm 2 can be terminated once $c_{\max} - c_{\min}$ is sufficiently small; in our experiments, 10-15 iterations usually sufficed.

5. Hardness of Bicriterion Approximation

Guarantees of the form presented in Theorem 5 are often called *bicriterion* guarantees. Instead of requiring that the obtained objective score is close to the optimal score *and all* constraints are exactly met, a bicriterion guarantee requires a bound on the suboptimality of the objective, as well as bounds on how much the constraints are violated. Theorem 3 showed that—unless P = NP—no approximation guarantees can be obtained which do not violate the constraint on the cost k, thereby necessitating the bricriterion analysis.

One might ask, whether the guarantee on the size of the set, α, can be improved. Unfortunately, this is not likely, as the following result shows:

Theorem 6 *If there were a polynomial time algorithm which, for any integer k, is guaranteed to find a solution A_S such that $\min_i F_i(A_S) \geq \max_{|A|\leq k} \min_i F_i(A)$ and $|A_S| \leq \beta k$, where $\beta \leq (1-\varepsilon)(1+\log\max_{s\in V}\sum_i F_i(\{s\}))$ for some fixed $\varepsilon > 0$, then* NP $\subseteq$ DTIME$(n^{\log\log n})$.

Hereby, DTIME($n^{\log\log n}$) is a class of deterministic, slightly superpolynomial (but sub-exponential) algorithms (Feige, 1998); the inclusion NP $\subseteq$ DTIME($n^{\log\log n}$) is considered unlikely (Feige, 1998). Taken together, Theorem 3 and Theorem 6, provide strong theoretical evidence that SATURATE achieves best possible theoretical guarantees for the problem of maximizing the minimum over a set of submodular functions.

6. Examples of Robust Submodular Observation Selection problems

We now demonstrate that many important machine learning problems can be phrased as RSOS problems. Section 8 provides more details and experimental results for these domains.

6.1 Minimizing the Maximum Kriging Variance

Consider a Gaussian Process (GP) (cf., Rasmussen and Williams, 2006) X_V defined over a finite set of locations (indices) V. Hereby, X_V is a set of random variables, one variable X_s for each location $s \in V$. Given a set of locations $A \subseteq V$ which we observe, we can compute the predictive distribution $P(X_{V\setminus A} \mid X_A = \mathbf{x}_A)$, that is, the distribution of the variables $X_{V\setminus A}$ at the unobserved locations $V \setminus A$, conditioned on the measurements at the selected locations, $X_A = \mathbf{x}_A$. Let $\sigma^2_{s|A}$ be the residual variance after making observations at A. Let Σ_{AA} be the covariance matrix of the measurements at the chosen locations A, and Σ_{sA} be the vector of cross-covariances between the measurements at s and A. Then, the predictive variance (often called Kriging variance in the geostatistics literature), given by

$$\sigma^2_{s|A} = \sigma^2_s - \Sigma_{sA}\Sigma^{-1}_{AA}\Sigma_{As},$$

depends only on the set A, and *not* on the observed values $\mathbf{x}_A$.[2] As argued in Section 2, an often (especially in the case of nonstationary phenomena) appropriate criterion is to select locations A such that the maximum marginal variance is as small as possible, that is, we want to select a subset $A^* \subseteq V$ of locations to observe such that

$$A^* = \operatorname*{argmin}_{|A|\leq k} \max_{s\in V} \sigma^2_{s|A}. \tag{7}$$

Let us assume for now that the a priori variance σ^2_s is constant for all locations s (in Section 7, we show how our approach generalizes to non-constant marginal variances). Furthermore, let us define the *variance reduction* $F_s(A) = \sigma^2_s - \sigma^2_{s|A}$. Solving Problem (7) is then equivalent to maximizing the minimum variance reduction over all locations s. For a particular location s, Das and Kempe (2008) show that the variance reduction F_s (often) is a monotonic submodular function. Hence the problem

$$A^* = \operatorname*{argmax}_{|A|\leq k} \min_{s\in V} F_s(A) = \operatorname*{argmax}_{|A|\leq k} \min_{s\in V} \sigma^2_s - \sigma^2_{s|A}$$

is an instance of the RSOS problem.

2. This independence is a particular property of the Gaussian distribution. When such independence is present, there is no benefit of sequentially selecting observations (cf., Krause and Guestrin, 2007b).

6.2 Variable Selection under Parameter Uncertainty

Consider an application, where we want to diagnose a failure of a complex system, by performing a number of tests. We can model this problem by using a set of discrete random variables $X_V = \{X_1, \ldots, X_n\}$ indexed by $V = \{1, \ldots, n\}$, which model both the hidden state of the system and the outcomes of the diagnostic tests. The interaction between these variables is modeled by a joint distribution $P(X_V \mid \theta)$ with parameters θ. Krause et al. (2007b) and Krause and Guestrin (2005) show that many variable selection problems can be formulated as the problem of optimizing a submodular utility function (measuring, for example, the information gain $I(X_U, X_A)$ with respect to some variables of interest U, or the mutual information $I(X_A; X_{V \setminus A})$ between the observed and unobserved variables, etc.). However, the informativeness of a chosen set A typically depends on the particular parameters θ, and these parameters might be uncertain. In some applications, it might not be reasonable to impose a prior distribution over θ, and we may want to perform well even under the worst-case parameters. In these cases, we can associate, with each parameter setting θ, a different submodular objective function F_θ, for example,

$$F_\theta(A) = I(X_A; X_U \mid \theta),$$

and we might want to select a set A which simultaneously performs well for all possible parameter values. In practice, we can discretize the set of possible parameter values θ (for example around a 95% confidence interval estimated from initial data) and optimize the worst case F_θ over the resulting discrete set of parameters.

6.3 Robust Experimental Designs

Another application is experimental design under nonlinear dynamics (Flaherty et al., 2006). The goal is to estimate a set of parameters θ of a nonlinear function $y = f(\mathbf{x}, \theta) + w$, by providing a set of experimental stimuli $\mathbf{x}$, and measuring the (noisy) response y. In many cases, experimental design for linear models (where $y = A(\mathbf{x})^T \theta + w$ with Gaussian noise w) can be efficiently solved by semidefinite programming (Boyd and Vandenberghe, 2004). In the nonlinear case, a common approach (cf., Chaloner and Verdinelli, 1995) is to *linearize* f around an initial parameter estimate θ_0, that is,

$$y = f(\mathbf{x}, \theta_0) + V(\mathbf{x})(\theta - \theta_0) + w, \tag{8}$$

where $V(\mathbf{x})$ is the Jacobian of f with respect to the parameters θ, evaluated at θ_0. Subsequently, a *locally-optimal* design is sought, which is optimal for the linear design Problem (8) for initial parameter estimates θ_0. Flaherty et al. (2006) show that the efficiency of such a locally optimal design can be very sensitive with respect to the initial parameter estimates θ_0. Consequently, they develop an efficient semi-definite program (SDP) for E-optimal design (i.e., the goal is to minimize the maximum eigenvalue of the error covariance) which is robust against perturbations of the Jacobian V. However, it might be more natural to directly consider robustness with respect to perturbation of the initial parameter estimates θ_0, around which the linearization is performed. We show how to find (Bayesian A-optimal) designs which are robust against uncertainty in these parameter estimates. In this setting, the objectives $F_{\theta_0}(A)$ are the reductions of the trace of the parameter covariance,

$$F_{\theta_0}(A) = \operatorname{tr}\left(\Sigma_\theta^{(\theta_0)}\right) - \operatorname{tr}\left(\Sigma_{\theta|A}^{(\theta_0)}\right),$$

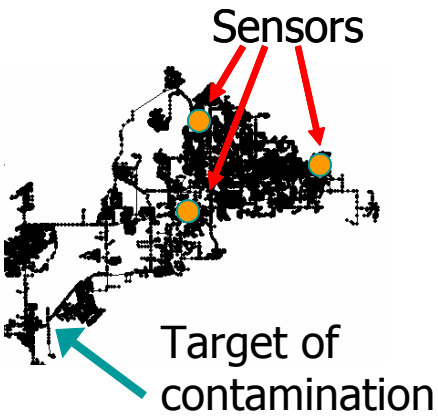


Figure 5: Securing a municipal water distribution network against contaminations performed under knowledge of the sensor placement is another instance of the RSOS problem.

where $\Sigma^{(\theta_0)}$ is the joint covariance of observations and parameters after linearization around θ_0; thus, F_{θ_0} is the sum of marginal parameter variance reductions, which are (often) individually monotonic and submodular (Das and Kempe, 2008), and so F_{θ_0} is monotonic and submodular as well. Hence, in order to find a robust design, we maximize the minimum variance reduction, where the minimum is taken over (a discretization into a finite subset of) all initial parameter values θ_0.

6.4 Sensor Placement for Outbreak Detection

Another class of examples are outbreak detection problems on graphs, such as contamination detection in water distribution networks (Leskovec et al., 2007). Here, we are given a graph $G = (V, E)$, and a phenomenon spreading dynamically over the graph. We define a set of *intrusion scenarios* I; each scenario $i \in I$ models an outbreak (e.g., spreading of contamination) starting from a given node $s \in V$ in the network. By placing sensors at a set of locations $A \subseteq V$, we can detect such an outbreak, and thereby minimize the adverse effects on the network.

More formally, for each possible outbreak scenario $i \in I$ and for each node $v \in V$ we define the detection time $T_i(v)$ as the time when the outbreak affects node v (and $T_i(v) = \infty$ if node v is never affected). We furthermore define a penalty function $\pi_i(t)$ which models the penalty incurred for detecting outbreak i at time t. We require $\pi_i(t)$ to be monotonically non-decreasing in t (i.e., we never prefer late over early detection), and bounded above by $\pi_i(\infty) \in \mathbb{R}$. Our goal is to minimize the worst-case penalty: We extend π_i to observation sets A as $\pi_i(A) = \pi_i(\min_{s \in A} T_i(s))$. Then, our goal is to solve

$$A^* = \operatorname*{argmin}_{|A| \leq k} \max_{i \in I} \pi_i(A).$$

Equivalently, we can define the *penalty reduction* $F_i(A) = \pi_i(\infty) - \pi_i(A)$. Clearly, $F_i(\emptyset) = 0$, F_i is monotonic. In Leskovec et al. (2007), it was shown that F_i is also guaranteed to be submodular. For now, let us assume that $\pi_i(\infty)$ is constant for all i (we will relax this assumption in Section 7.2). Our goal in sensor placement is then to select a set of sensors A such that the minimum penalty

reduction is as large as possible, that is, we want to select

$$A^* = \operatorname*{argmax}_{|A|\leq k} \min_{i\in I} F_i(A).$$

In other words, an adversary observes our sensor placement A, and then decides on an intrusion i for which our utility $F_i(A)$ is as small as possible. Hence, our goal is to find a placement A which performs well against such an adversarial opponent.

6.5 Robustness Against Sensor Failures and Feature Deletion

Another interesting instance of the RSOS problem arises in the context of robust sensor placements. For example, in the outbreak detection problem, sensors might *fail*, due to hardware problems or manipulation by an adversary. We can model this problem in the following way: Consider the case where all sensors at a subset $B \subseteq V$ of locations fail. Given a submodular function F (e.g., the utility for placing a set of sensors), and the set $B \subseteq V$ of failing sensors, we can define a new function $F_B(A) = F(A \setminus B)$, corresponding to the (reduced) utility of placement A after the sensor failures. It is easy to show that if F is nondecreasing and submodular, so is F_B. Hence, the problem of optimizing sensor placements which are robust to sensor failures results in a problem of simultaneously maximizing a collection of submodular functions, for example, for the worst-case failure of $k' < k$ sensors we solve

$$A^* = \operatorname*{argmax}_{|A|\leq k} \min_{|B|\leq k'} F_B(A).$$

We can also combine the optimization against adversarial contamination scenarios as discussed in Section 6.3 with adversarial sensor failures, and optimize[3]

$$A^* = \operatorname*{argmax}_{|A|\leq k} \min_{i\in I} \min_{|B|\leq k'} F_i(A \setminus B).$$

Another important problem in machine learning is feature selection. In feature selection, the goal is to select a subset of features which are informative with respect to, for example, a given classification task. One objective frequently considered is the problem of selecting a set of features which maximize the information gained about the class variable X_Y after observing the features X_A, $F(A) = H(X_Y) - H(X_Y \mid X_A)$, where H denotes the Shannon entropy. Krause and Guestrin (2005) show, that in a large class of graphical models, the information gain $F(A)$ is in fact a submodular function. Now we can consider a setting, where an adversary can delete features which we selected (as considered, for example, by Globerson and Roweis 2006). The problem of selecting features robustly against such arbitrary deletion of, for example, m features, is hence equivalent to the problem of maximizing $\min_{|B|\leq m} F_B(A)$, where B are the deleted features.

3. Note that for larger values of k', computing the average truncated utility can be computationally complex. See Section 9 for possible approaches to reduce this complexity. Also note that in practice, one can expect that sensor failures have structure (e.g., sensors that are spatially collocated, or share other common features, are more likely to simultaneously fail). Such structured failures can potentially be modeled by appropriately choosing the collection of sets B of failing nodes.

6.5.1 IMPROVED GUARANTEES FOR SENSOR FAILURES

As discussed above, in principle, we could find a placement robust to single sensor failures by using SATURATE to (approximately) solve

$$A^* = \operatorname*{argmax}_{|A|\leq k} \min_s F_s(A).$$

However, since $|V|$ can be very large, and the approximation guarantee α depends logarithmically on $|V|$, such a direct approach might not be desirable. We can improve the guarantee from $O(\log|V|)$ to $O(\log(k\log|V|))$, which typically is much tighter, if $k \ll |V|/\log|V|$ (i.e., we place far fewer sensors than we have possible sensor locations). We can improve the approximation guarantee drastically by noticing that $F_s(A) = F(A)$ if $s \notin A$. Hence,

$$\overline{F}_c(A) = \frac{|V|-|A|}{|V|}\min\{F(A),c\} + \frac{1}{|V|}\sum_{s\in A}\widehat{F}_{s,c}(A).$$

We can replace this objective by a new objective function,

$$\overline{F}'_c(A) = \frac{k'-|A|}{k'}\min\{F(A),c\} + \frac{1}{k'}\sum_{s\in A}\widehat{F}_{s,c}(A)$$

for some constant $\hat{k}$ to be specified below. This modified objective is still monotonic and submodular when restricted to sets of size at most $\hat{k}$. It still holds that, for all subsets $|A| \leq \hat{k}$, that

$$\overline{F}'_c(A) \geq c \Leftrightarrow F_s(A) \geq c \text{ for all } s \in V.$$

How large should we choose $\hat{k}$? We have to choose $\hat{k}$ large enough such that SATURATE will never choose sets larger than $\hat{k}$. A sufficient choice for $\hat{k}$ is hence $\lceil \alpha k \rceil$, where $\alpha = 1+\log(|V|\max_{s\in V} F(\{s\}))$. For this choice of $\hat{k}$, our new approximation guarantee will be

$$\begin{aligned}\alpha' &= 1+\log\left(\alpha k \max_{s\in V} F(\{s\})\right) = 1+\log\left(\left(1+\log\left(|V|\max_{s\in V}F(\{s\})\right)\right)k\max_{s\in V}F(\{s\})\right)\\ &\leq 1+2\log\left(k\log(|V|)\max_{s\in V}F(\{s\})\right)\end{aligned}$$

Hence, for the new objective $\overline{F}'_c$, we get a tighter approximation guarantee, $\alpha' = 1+2\log(k\log(|V|)\max_{s\in V}F(\{s\}))$, which now depends logarithmically on $k\log|V|$, instead of the number of available locations $|V|$. Note that this same approach can also provide tighter approximation guarantees in the case of multiple sensor failures.

7. Extensions

We now show how some of the assumptions made in our presentation above can be relaxed. We also discuss several extensions, allowing more complex cost functions, and the tradeoff between worst-case and average-case scores.

7.1 Non-integral Objectives

In our analysis of SATURATE (Section 4), we have assumed, that each of the objective functions F_i only take values in the positive integers. However, most objective functions of interest in observation selection (such as those discussed in Section 6) typically do not meet this assumption. If the F_i take on rational numbers, we can scale the objectives by multiplying by their common denominator.

If we allow small additive approximation error (i.e., are indifferent if the approximate solution differs from the optimal solution in low order bits), we can also approximate the values assumed by the functions F_i by their highest order bits. In this case, we replace the functions $F_i(A)$ by the approximations

$$F_i'(A) = \frac{\lceil 2^j F_i(A) \rceil}{2^j}.$$

By construction, $F_i'(A) \leq F_i(A) \leq F_i'(A)(1+2^{-j})$, that is, F_i' is within a factor of $(1+2^{-j})$ of F_i. Also, $2^j F_i'(A)$ is integral. However, $F_i'(A)$ is not guaranteed to be submodular. Nevertheless, an analysis similar to the one presented by Krause et al. (2007b) can be used to bound the effect of this approximation on the theoretical guarantees α obtained by the algorithm, which will now scale linearly with the number j of high order bits considered. In practice, as we show in Section 8, SATURATE provides state-of-the-art performance, even without rounding the objectives to the highest order bits.

7.2 Non-constant Thresholds

Consider the example of minimizing the maximum variance in Gaussian Process regression. Here, the $F_i(A) = \sigma_i^2 - \sigma_{i|A}^2$ denote the variance reductions at location i. However, rather than guaranteeing that $F_i(A) \geq c$ for all i (which, in this example, means that the *minimum variance reduction* is c), we want to guarantee that $\sigma_{i|A}^2 \leq c$ for all i, which requires a different amount of variance reduction for each location. We can easily adapt our approach to handle this case: Instead of defining $\widehat{F}_{i,c}(A) = \min\{F_i(A), c\}$, we define $\widehat{F}_{i,c}(A) = \min\{F_i(A), \sigma_i^2 - c\}$, and then again perform binary search over c, but searching for the smallest c instead. The algorithm, using objectives modified in this way, will bear the same approximation guarantees.

7.3 Non-uniform Observation Costs

We can extend SATURATE to the setting where different observations have different costs. In the spatial monitoring setting for example, certain locations might be more expensive to acquire a measurement from. Suppose a cost function $g : V \to \mathbb{R}^+$ assigns each element $s \in V$ a positive cost $g(s)$; the cost of a set of observations is then $g(A) = \sum_{s \in A} g(s)$. The problem is to find $A^* = \operatorname{argmax}_{A \subset V} \min_i F_i(A)$ subject to $g(A) \leq B$, where $B > 0$ is a *budget* we can spend on making observations. In this case, we use the rule

$$\delta_s \leftarrow \frac{\overline{F}_c(A \cup \{s\}) - \overline{F}_c(A)}{g(s)}$$

in Algorithm 1. For this modified algorithm, Theorem 5 still holds, with $|A|$ replaced by $g(A)$ and k replaced by B. This more general result holds, since the analysis of the greedy algorithm for submodular covering of Wolsey (1982), which we used to prove Lemma 4, applies to the more general setting of non-uniform cost functions.

7.4 Handling More Complex Cost Functions

So far, we considered problems where we are given an *additive* cost function $g(A)$ over the possible sets A of observations. In some applications, more complex cost functions arise. For example, when placing wireless sensor networks, the placements A should not only be informative (i.e., $F_i(A)$ should be high for all utility functions F_i), but the placement should also have *low communication cost*. Krause et al. (2006) describe such an approach, where the cost $g(A)$ measures the *expected number of retransmissions* required for sending messages across an optimal routing tree connecting the sensors A. Formally, the observations s are considered to be nodes in a graph $G = (V, E)$, with edge weights $w(e)$ for each edge $e \in E$. The cost $g(A)$ is the cost of a minimum Steiner Tree (cf., Vazirani 2003) connecting the observations A in the graph G.

More generally, we want to solve problems of the form

$$\operatorname*{argmax}_A \min_i F_i(A) \text{ subject to } g(A) \leq B,$$

where $g(A)$ is a complex cost function. The key insight of the SATURATE algorithm is that the non-submodular robust optimization problem can be approximately solved by solving a submodular covering problem. In the case where $g(A) = |A|$ this problem requires solving (6). More generally, we can apply SATURATE to any problem where we can (approximately) solve

$$A_c = \operatorname*{argmin}_{A \subseteq V} g(A), \quad \text{such that} \quad \overline{F}_c(A) = c. \tag{9}$$

Problem (9) can be (approximately) solved for a variety of cost functions, such as those arising from communication constraints (Krause et al., 2006) and path constraints (Singh et al., 2007; Meliou et al., 2007).

Let us summarize our analysis as follows:

Proposition 7 *Assume we have an algorithm which, given a monotonic submodular function F and a cost function g, returns a solution A' such that $F(A') = F(V)$ and*

$$g(A') \leq \alpha_F \min_{A: F(A) = F(V)} g(A),$$

where α_F depends on the function F. SATURATE*, using this covering algorithm, can obtain a solution A_S to the* RSOS *problem such that*

$$\min_i F_i(A_S) \geq \max_{g(A) \leq B} \min_i F_i(A),$$

and

$$g(A_S) \leq \alpha_{\overline{F}} B,$$

where $\alpha_{\overline{F}}$ is the approximation factor of the covering algorithm, when applied to $\overline{F} = \frac{1}{m}\sum_i F_i$.

Note that the formalism developed in this section also allows to handle robust versions of combinatorial optimization problems such as the *Knapsack* (cf., Martello and Toth, 1990), Orienteering (cf., Laporte and Martello, 1990; Blum et al., 2003) and *Budgeted Steiner Tree* (cf., Johnson et al., 2000) problems. In these problems, instead of a general *submodular* objective function, the special case of a *modular* (additive) function F is optimized:

$$A^* = \operatorname*{argmax}_{g(A) \leq B} F(A).$$

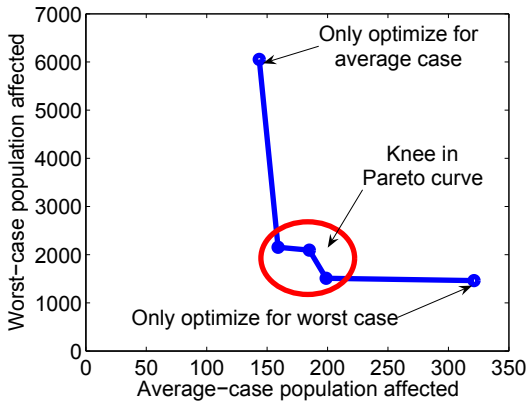


Figure 6: Tradeoff curve for simultaneously optimizing the average- and worst-case score in the water distribution network monitoring application. Notice the knee in the tradeoff curve, indicating that by performing multi-criterion optimization, solutions performing well for both average- and worst-case scores can be obtained.

The problems differ only in the choice of the complex cost function. In *Knapsack* for example, g is additive, in the *Budgeted Steiner Tree* problem, $g(A)$ is the cost of a minimum Steiner tree connecting the nodes A in a graph, and in *Orienteering*, $g(A)$ is the cost of a shortest path connecting the nodes A in a graph. In practice, often the utility function $F(A)$ is not exactly known, and a solution is desired which is robust against worst-case choice of the utility function. Since modular functions are a special case of submodular functions, such problems can be approximately solved using Proposition 7.

7.5 Trading Off Average-case and Worst-case Scores

In some applications, optimizing the worst-case score $F_{wc}(A) = \min_i F_i(A)$ might be a too pessimistic approach. On the other hand, ignoring the worst-case and only optimizing the average-case (the expected score under a distribution over the objectives) $F_{ac}(A) = \frac{1}{m}\sum_i F_i(A)$ might be too optimistic. In fact, in Section 8 we show that optimizing the average-case score F_{ac} can often lead to drastically poor worst-case scores. In general, we might be interested in solutions, which perform well both in the average- and worst-case scores.

Formally, we can define a multicriterion optimization problem, where we intend to optimize the *vector* $[F_{ac}(A), F_{wc}(A)]$. In this setting, we can only hope for *Pareto-optimal* solutions (cf., Boyd and Vandenberghe, 2004, in the context of convex functions). A set A^*, $|A^*| \leq k$ is called *Pareto-optimal*, if it is not *dominated*, that is, there does not exist another set B, $|B| \leq k$ with $F_{ac}(B) > F_{ac}(A^*)$ and $F_{wc}(B) \geq F_{wc}(A^*)$ (or $F_{ac}(B) \geq F_{ac}(A^*)$ and $F_{wc}(B) > F_{wc}(A^*)$).

One possible approach to find such Pareto-optimal solutions is constrained optimization:[4] for a specified value of c_{ac}, we desire a solution to

$$A^* = \operatorname*{argmax}_{|A|\leq k} F_{wc}(A) \text{ such that } F_{ac}(A) \geq c_{ac}. \tag{10}$$

4. Another approach is *scalarization*, where we optimize $F_\lambda(A) = \lambda F_{wc}(A) + (1-\lambda)F_{ac}(A)$ for some λ, $0 < \lambda < 1$. SATURATE can be modified to handle such scalarized objectives as well.

By specifying different values of c_{ac} in (10), we would obtain different Pareto-optimal solutions.[5] Figure 6 presents an example of several Pareto-optimal solutions, based on data from the outbreak detection problem (Details will be discussed in Section 8.3). This curve shows that, using the techniques described below, multicriterion solutions can be found which combine the advantages of worst-case and average-case solutions.

We can modify SATURATE to solve Problem (10) in the following way. Let us again assume we know the optimal value c_{wc} achievable for Problem (10). Then, Problem (10) is equivalent to solving

$$A^* = \operatorname*{argmin}_A |A| \text{ subject to } F_{wc}(A) \geq c_{wc} \text{ and } F_{ac}(A) \geq c_{ac}.$$

Now, using our notation from Section 4, this problem is again equivalent to

$$A^* = \operatorname*{argmin}_A |A| \text{ subject to } \overline{F}_{c_{wc},c_{ac}} = c_{wc} + c_{ac}, \tag{11}$$

where

$$\overline{F}_{c_{wc},c_{ac}}(A) = \overline{F}_{c_{wc}}(A) + \min\{F_{ac}(A), c_{ac}\}.$$

Note that $\overline{F}_{c_{wc},c_{ac}}$ is a submodular function, and hence (11) is a submodular covering problem, which can be approximately solved using the greedy algorithm.

For any choice of c_{ac}, we can find the optimal value of c_{wc} by performing binary search on c_{wc}. We summarize our analysis in the following Theorem:

Theorem 8 *For any integer k and constraint c_{ac},* SATURATE *finds a solution A_S (if it exists) such that*

$$F_{wc}(A_S) \geq \max_{|A| \leq k, F_{ac}(A) \geq c_{ac}} F_{wc}(A),$$

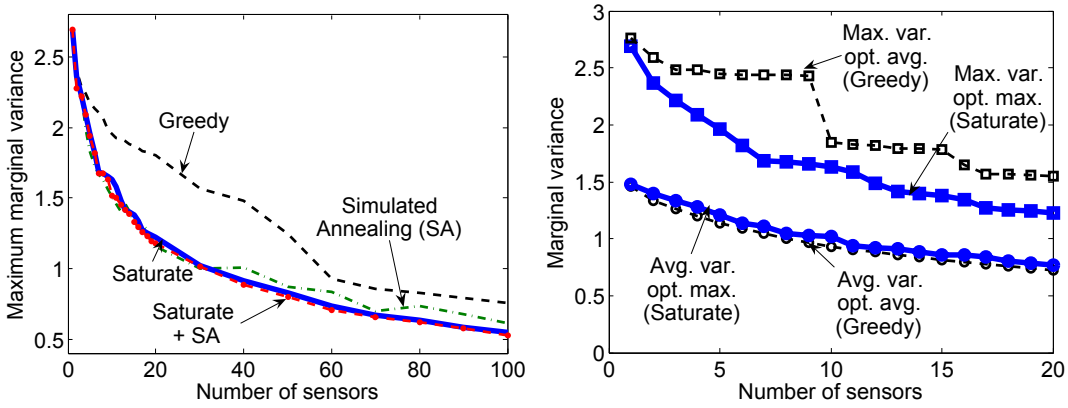
$F_{ac}(A_S) \geq c_{ac}$, and $|A_S| \leq \alpha k$, for $\alpha = 1 + \log(2 \max_{s \in V} \sum_i F_i(\{s\}))$. Each such solution A_S is approximately *Pareto-optimal, that is, there does not exist a set B, $|B| \leq k$ such that B dominates A_S. The total number of submodular function evaluations is $O\left(|V|^2 m \log(\sum_i F_i(V))\right)$.*

8. Experimental Results

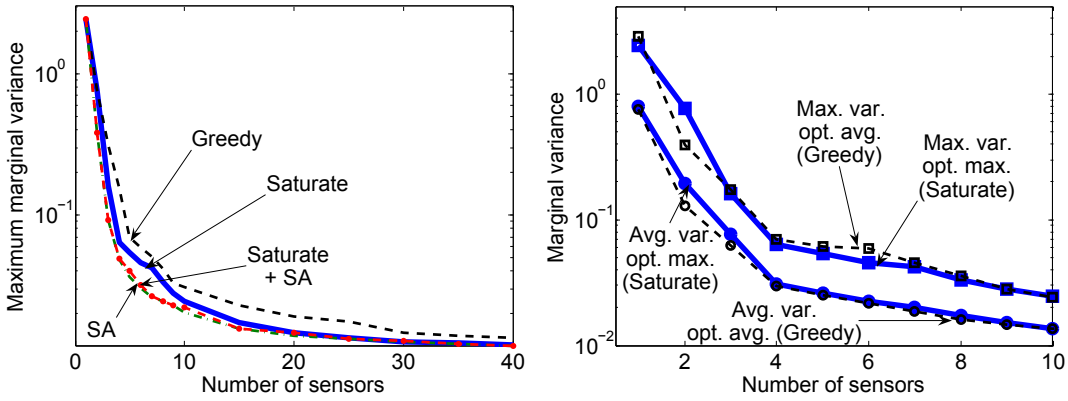
In this section, we present experimental results on several robust observation selection problems.

8.1 Minimizing the Maximum Kriging Variance

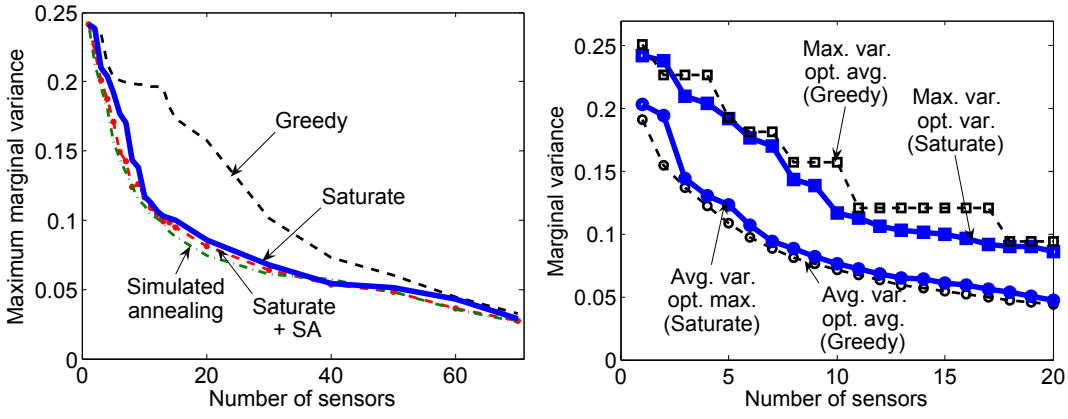
First, we use SATURATE to select observations in a GP to minimize the maximum posterior variance (cf., Section 6.1). We consider three data sets: [T] temperature data from a deployment of 52 sensors at Intel Research Berkeley, [P] Precipitation data from the Pacific Northwest of the United States (Widmann and Bretherton, 1999) and [L] temperature data from the NIMS sensor node (Harmon et al., 2006) deployed at a lake near the University of California, Merced. For the three monitoring problems, [T], [P], and [L], we discretize the space into 46, 167 and 86 locations each, respectively. For [T], we consider the empirical covariance matrix of temperature sensor measurements obtained over a period of 5 days. For [P], we consider the empirical covariance of 50 years of data, which we preprocessed as described by Krause et al. (2007b). For [L], we train a nonstationary Gaussian Process using data from a single scan of the lake by the NIMS sensor node, using a method described by Krause and Guestrin (2007b).



(a) *[P] Algorithm comparison* (b) *[P] Avg. vs max. variance*



(c) *[T] Algorithm comparison* (d) *[T] Avg. vs max. variance*



(e) *[L] Algorithm comparison* (f) *[L] Avg. vs max. variance*

Figure 7: (a,c,e) SATURATE, greedy and SA on the (a) precipitation, (b) building temperature and (c) lake temperature data. SATURATE performs comparably with the fine-tuned SA algorithm, and outperforms it for larger placements. (b,d,f) Optimizing for the maximum variance (using SATURATE) leads to low average variance, but optimizing for average variance (using greedy) does not lead to low maximum variance.

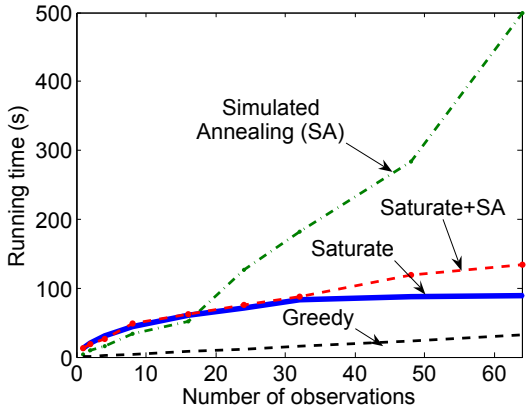


Figure 8: Running time for algorithms on the precipitation data set [P].

In the geostatistics literature, the predominant choice of optimization algorithms for selecting locations in a GP to minimize the (maximum and average) predictive variance are carefully tuned local search procedures, prominently simulated annealing (cf., Sacks and Schiller 1988; Wiens 2005; van Groenigen and Stein 1998). We compare our SATURATE algorithm against a state-of-the-art implementation of such a simulated annealing (SA) algorithm, first proposed by Sacks and Schiller (1988). We use an optimized implementation described recently by Wiens (2005). This algorithm has 7 parameters which need to be tuned, describing the annealing schedule, distribution of iterations among several inner loops, etc. We use the parameter settings as reported by Wiens (2005), and present the best result of the algorithm among 10 random trials. In order to compare observation sets of the same size, we called SATURATE with $\alpha = 1$.

Figures 7(a), 7(c) and 7(e) compare simulated annealing, SATURATE, and the greedy algorithm which greedily selects elements which decrease the maximum variance the most on the three data sets. We also used SATURATE to initialize the simulated annealing algorithm (using only a single run of simulated annealing, as opposed to 10 random trials). In all three data sets, SATURATE obtains placements which are drastically better than the placements obtained by the greedy algorithm. Furthermore, the performance is very close to the performance of the simulated annealing algorithm. In our largest monitoring data set [P], SATURATE even strictly outperforms the simulated annealing algorithm when selecting 30 and more sensors. Furthermore, as Figure 8 shows, SATURATE is significantly faster than simulated annealing, by factors of 5-10 for larger problems. When using SATURATE in order to initialize the simulated annealing algorithm, the resulting performance almost always resulted in the best solutions we were able to find with any method, while still executing faster than simulated annealing with 10 random restarts as proposed by Wiens (2005). These results indicate that SATURATE compares favorably to state-of-the-art local search heuristics, while being faster, requiring no parameters to tune, and providing theoretical approximation guarantees.

Optimizing for the maximum variance could potentially be considered too pessimistic. Hence we compared placements obtained by SATURATE, minimizing the maximum marginal posterior variance, with placements obtained by the greedy algorithm, where we minimize the *average* marginal variance. Note, that, whereas the maximum variance reduction is non-submodular, the *average* variance reduction is (often) submodular (Das and Kempe, 2008), and hence the greedy algorithm can

5. In fact, *all* Pareto-optimal solutions can be found in this way (Papadimitriou and Yannakakis, 2000).

be expected to provide near-optimal placements. Figures 7(b), 7(d) and 7(f) present the maximum and average marginal variances for both algorithms. On all three data sets, our results show that if we optimize for the maximum variance we still achieve comparable average variance. If we optimize for average variance however, the maximum posterior variance remains much higher.

8.2 Robust Experimental Design

We consider the robust design of experiments (cf., Section 6.3) for the Michaelis-Menten mass-action kinetics model, as discussed by Flaherty et al. (2006). The goal is least-square parameter estimation for a function $y = f(x,\theta)$, where x is the chosen experimental stimulus (the initial substrate concentration S_0), and $\theta = (\theta_1,\theta_2)$ are two parameters as described by Flaherty et al. (2006). The stimulus x is chosen from a menu of six options, $x \in \{1/8,1,2,4,8,16\}$, each of which can be repeatedly chosen. The goal is to produce a fractional design $\mathbf{w} = (w_1,\ldots,w_6)$, where each component w_i measures the relative frequency according to which the stimulus x_i is chosen. Since f is nonlinear, f is linearized around an initial parameter estimate $\theta_0 = (\theta_{01},\theta_{02})$, and approximated by its Jacobian V_{θ_0}. Classical experimental design considers the error covariance of the least squares estimate $\hat{\theta}$, $\mathrm{Cov}(\hat{\theta} \mid \theta_0,\mathbf{w}) = \sigma^2(V_{\theta_0}^T W V_{\theta_0})^{-1}$, where $W = \mathrm{diag}(\mathbf{w})$, and aims to find designs $\mathbf{w}$ which minimize this error covariance. E-optimality, the criterion adopted by Flaherty et al. (2006), measures smallness in terms of the maximum eigenvalue of the error covariance matrix. The optimal $\mathbf{w}$ can be found using Semidefinite Programming (SDP) (Boyd and Vandenberghe, 2004).

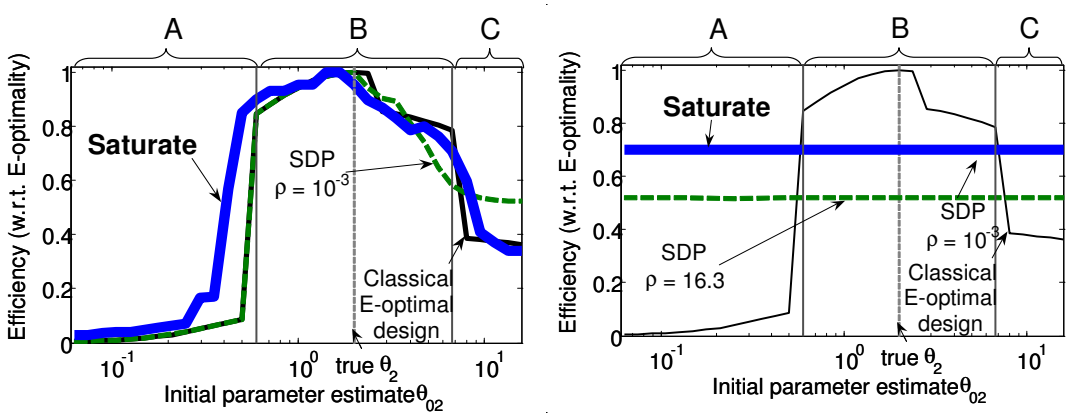
The estimate $\mathrm{Cov}(\hat{\theta} \mid \theta_0,\mathbf{w})$ depends on the initial parameter estimate θ_0, where linearization is performed. However, since the goal is parameter estimation, a "certain circularity is involved" (Flaherty et al., 2006). To avoid this problem, Flaherty et al. (2006) find a design $\mathbf{w}_\rho(\theta_0)$ by solving a robust SDP which minimizes the error size, subject to a worst-case perturbation Δ on the Jacobian V_{θ_0}; the robustness parameter ρ bounds the spectral norm of Δ. As evaluation criterion, Flaherty et al. (2006) define a notion of *efficiency*, which is the error size of the optimal design with correct initial parameter estimate, divided by the error when using a robust design obtained at the wrong initial parameter estimates, that is,

$$\text{efficiency} \equiv \frac{\lambda_{\max}[\mathrm{Cov}(\hat{\theta} \mid \theta_{true}, \mathbf{w}_{opt}(\theta_{true}))]}{\lambda_{\max}[\mathrm{Cov}(\hat{\theta} \mid \theta_{true}, \mathbf{w}_\rho(\theta_0))]},$$

where $\mathbf{w}_{opt}(\theta)$ is the E-optimal design for parameter θ. They show that for appropriately chosen values of ρ, the robust design is more *efficient* than the optimal design, if the initial parameter θ_0 does not equal the true parameter.

While their results are very promising, an arguably more natural approach than perturbing the Jacobian would be to perturb the initial parameter estimate, around which linearization is performed. For example, if the function f describes a process which behaves characteristically differently in different "phases", and the parameter θ controls which of the phases the process is in, then a robust design should intuitively "hedge" the design against the behavior in each possible phase. In such a case, the uniform distribution (which the robust SDP chooses for large ρ) would not be the most robust design.

If we discretize the space of possible parameter perturbations (within a reasonably chosen interval), we can use SATURATE to find robust experimental designs. While the classical E-optimality is not submodular (Krause et al., 2007b), Bayesian A-optimality is (usually) submodular (Das and Kempe, 2008; Krause et al., 2007b). Here, the goal is to minimize the *trace* instead of maximum



(a) *Low uncertainty in* θ_0 (b) *High uncertainty in* θ_0

Figure 9: Efficiency of robust SDP of Flaherty et al. (2006) and SATURATE on a biological experimental design problem. (a) Low assumed uncertainty in initial parameter estimates: SDP performs better in region C, SATURATE performs better in region A. (b) High assumed uncertainty in initial parameter estimates: SATURATE outperforms the SDP solutions.

eigenvalue size of the covariance matrix. Furthermore, we equip the parameters θ with an uninformative normal prior (which we chose as $\mathrm{diag}([20^2, 20^2])$) as typically done in Bayesian experimental design. We then minimize the expected trace of the posterior error covariance, $\mathrm{tr}(\Sigma_{\theta|A})$. Hereby, A is a discrete design of 20 experiments, where each option x_i can be chosen repeatedly. In order to apply SATURATE, for each θ_0, we define $F_{\theta_0}(A)$ as the normalized variance reduction

$$F_{\theta_0}(A) = \frac{1}{Z_{\theta_0}} \left(\mathrm{tr}\left(\Sigma_{\theta}^{(\theta_0)}\right) - \mathrm{tr}\left(\Sigma_{\theta|A}^{(\theta_0)}\right) \right).$$

The normalization Z_{θ_0} is chosen such that $F_{\theta_0}(A) = 1$ if

$$A = \operatorname*{argmax}_{|A'|=20} F_{\theta_0}(A'),$$

that is, if A is chosen to maximize only F_{θ_0}. SATURATE is then used to maximize the worst-case normalized variance reduction.

We reproduced the experiment of Flaherty et al. (2006), where the initial estimate of the second component θ_{02} of θ_0 was varied between 0 and 16, the "true" value being $\theta_2 = 2$. For each initial estimate of θ_{02}, we computed a robust design, using the SDP approach and using SATURATE, and compared them using the efficiency metric of Flaherty et al. (2006). Note that this efficiency metric is defined with respect to E-optimality, even though we optimize Bayesian A-optimality, hence potentially putting SATURATE at a disadvantage. We first optimized designs which are robust against a small perturbation of the initial parameter estimate. For the SDP, we chose a robustness parameter $\rho = 10^{-3}$, as reported in Flaherty et al. (2006). For SATURATE, we considered an interval around $[\theta\frac{1}{1+\varepsilon}, \theta(1+\varepsilon)]$, discretized in a 5×5 grid, with $\varepsilon = .1$.

Figure 9(a) shows three characteristically different regions, *A*, *B*, *C*, separated by vertical lines. In region *B* which contains the true parameter setting, the E-optimal design (which is optimal if

the true parameter is known, that is, $\theta_{02} = \theta_2$) performs similar to both robust methods. Hence, in region B (i.e., small deviation from the true parameter), robustness is not really necessary. Outside of region B however, where the standard E-optimal design performs badly, both robust designs do not perform well either. This is an intuitive result, as they were optimized to be robust only to small parameter perturbations.

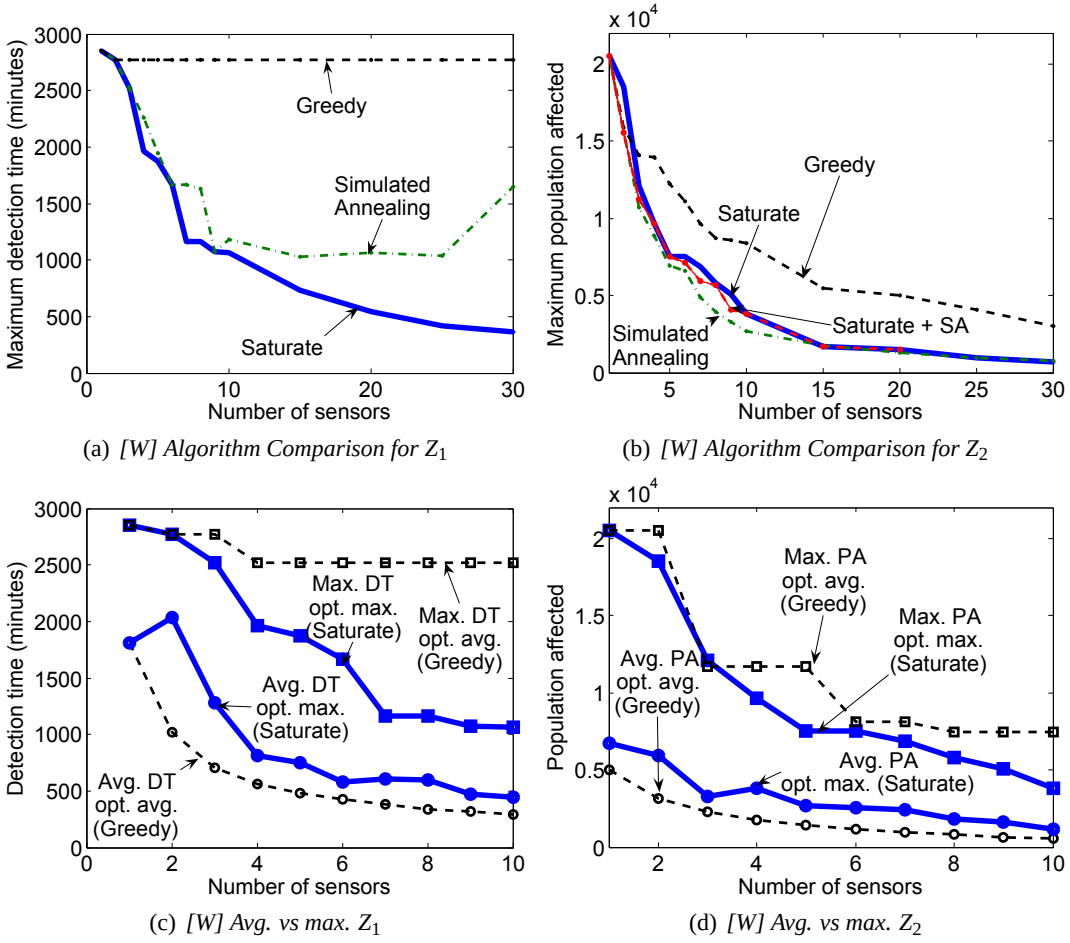
Consequently, we compared designs which are robust against a *large* parameter range. For SDP, we chose $\rho = 16.3$, which is the maximum spectral variation of the Jacobian when we consider all initial estimates from θ_{02} varying between 0 and 16. For SATURATE, we optimized a single design which achieves the maximum normalized variance reduction over all values of θ_{02} between 0 and 16. Figure 9(b) shows, that in this case, the design obtained by SATURATE achieves an efficiency of 69%, whereas the efficiency of the SDP design is only 52%. In the regions A and C, the SATURATE design strictly outperforms the other robust designs. This experiment indicates that designs which are robust against a large range of initial parameter estimates, as provided by SATURATE, can be more efficient than designs which are robust against perturbations of the Jacobian (the SDP approach).

8.3 Outbreak Detection

Consider a city water distribution network, delivering water to households via a system of pipes, pumps, and junctions. Accidental or malicious intrusions can cause contaminants to spread over the network, and we want to select a few locations (pipe junctions) to install sensors, in order to detect these contaminations as quickly as possible (cf., Section 6.3). In August 2006, the Battle of Water Sensor Networks (BWSN) (Ostfeld et al., 2006) was organized as an international challenge to find the best sensor placements for a real (but anonymized) metropolitan water distribution network, consisting of 12,527 nodes. In this challenge, a set of intrusion scenarios is specified, and for each scenario a realistic simulator provided by the EPA (Rossman, 1999) is used to simulate the spread of the contaminant for a 48 hour period. An intrusion is considered detected when one selected node shows positive contaminant concentration. BWSN considered a variety of impact measures, including the time to detection (called Z_1), and the size of the affected population calculated using a realistic disease model (Z_2). The goal of BWSN was to minimize the *expectation* of the impact measures Z_1 and Z_2 given a *uniform distribution* over intrusion scenarios.

In this paper, we consider the *adversarial* setting, where an opponent chooses the contamination scenario with knowledge of the sensor locations. The objective functions Z_1 and Z_2 are in fact submodular for a fixed intrusion scenario (Leskovec et al., 2007), and so the robust optimization problem of minimizing the impact of the worst possible intrusion fits into our formalism. For these experiments, we consider scenarios which affect at least 10% of the network, resulting in a total of 3424 scenarios. Figures 10(a) and 10(b) compare the greedy algorithm, SATURATE and the simulated annealing (SA) algorithm for the problem of maximizing the worst-case detection time (Z_1) and worst-case affected population (Z_2).

Interestingly, the behavior is very different for the two objectives. For the affected population (Z_2), greedy performs reasonably, and SA sometimes even outperforms SATURATE. For the detection time (Z_1), however, the greedy algorithm did not improve the objective at all, and SA performs poorly. The reason is that for Z_2, the maximum achievable scores, $F_i(V)$, vary drastically, since some scenarios have much higher impact than others. Hence, there is a strong "gradient", as the worst-case objective changes quickly when the high impact scenarios are covered. This gradient



(a) *[W] Algorithm Comparison for* Z_1

(b) *[W] Algorithm Comparison for* Z_2

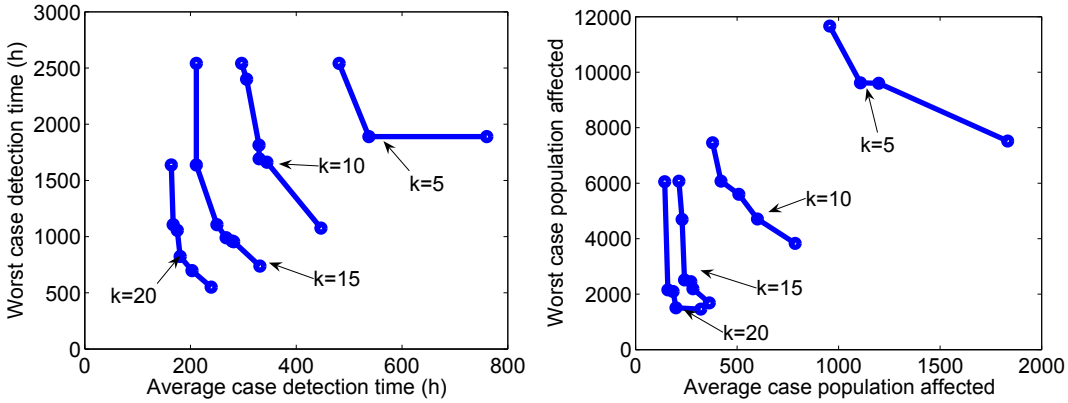
(c) *[W] Avg. vs max.* Z_1

(d) *[W] Avg. vs max.* Z_2

Figure 10: (a,b) compare SATURATE, greedy and SA in the water network setting, when optimizing worst-case detection time (Z_1, (a)) and affected population (Z_2, (b)). SATURATE performs comparably to SA for Z_2 and strictly outperforms SA for Z_1. (c,d) compare optimizing for the worst-case vs. average-case objectives. Optimizing for the worst-case leads to good average case performs, but not vice versa.

allows greedy and SA to work well. On other hand, for Z_1, the maximum achievable scores, $F_i(V)$, are constant, since all scenarios have the same simulation duration. Unless *all* scenarios are detected, the worst-case detection time stays constant at the simulation length. Hence, many node exchange proposals considered by SA, as well as the addition of a new sensor location by greedy, do not change the worst-case objective, and the algorithms have no useful performance metric.

Figures 10(c) and 10(d) compare the placements of SATURATE (when optimizing the worst-case penalty), and greedy (when optimizing the average-case penalty, which is submodular). Similarly to the results in the GP setting, optimizing the worst-case score leads to reasonable performance in the average case score, but not necessarily vice versa (especially when considering the detection time).



(a) *[W] Pareto curve for Z_1*

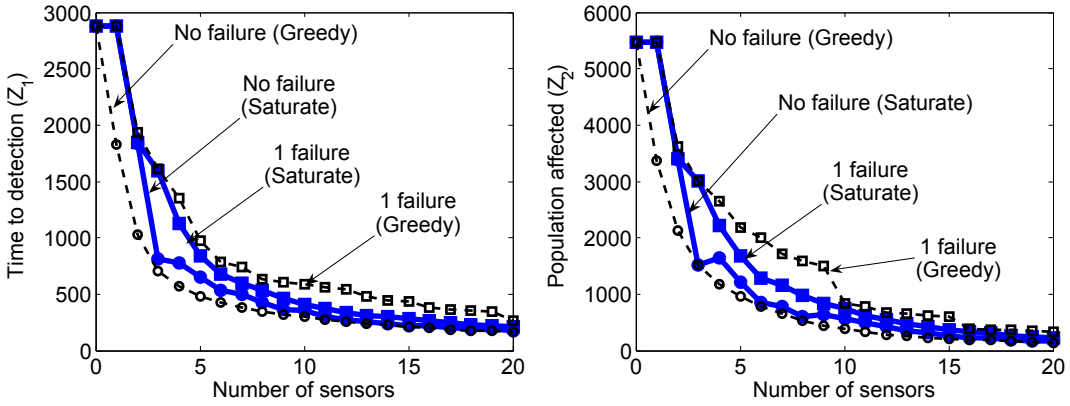
(b) *[W] Pareto curve for Z_2*

Figure 11: Experiments on trading off worst-case and average-case penalties on the water network [W] data, minimizing detection time (a) and affected population (b).

We also performed experiments trading off the worst-case and average-case penalty reductions, using the approach discussed in Section 7.5. We first ran the greedy algorithm to optimize the average-case score, and then ran SATURATE to optimize the worst-case score. We considered the average-case scores c_{ac}^{greedy} and $c_{ac}^{Saturate}$ obtained by both algorithms, and uniformly discretized the interval bounded by these average-case scores. For each score level c_{ac} in the discretization, we use the modified SATURATE algorithm as described in Section 7.5, maximizing the worst-case score, subject to a constraint on the average-case score. Each possible value of the constraint on c_{ac} can lead to a different solution, trading off average- and worst-case scores. Figure 11(a) presents the tradeoff curve obtained in this fashion for the detection time (Z_1) metric, for different numbers k of placed sensors. We generally observe that there is more variability in the worst-case score than in the average-case score. We can also see that when placing 5 sensors, there is a prominent knee in the tradeoff curve, effectively achieving the minimum worst-case penalty but drastically reducing the average-case penalty incurred when compared to only optimizing for the worst-case score. The other tradeoff curves do not exhibit quite such prominent knees, but nevertheless allow flexibility in trading off worst- and average-case scores. Figure 11(b) presents the same experiment for the population affected (Z_2) metric. Here, we notice prominent knees when placing $k = 15$ and 20 sensors. We can generally conclude that trading off average- and worst-case scores allows to effectively achieve a compromise between too pessimistic (only optimizing for the worst case) and optimistic (only optimizing for the average case) objectives.

8.4 Sensor Failures

We also performed experiments on analyzing worst-case sensor failures (cf., Section 6.5). We consider the outbreak detection application, and optimize the average score, that is, $F(A) = \frac{1}{m}\sum_i F_i(A)$ (modeling, for example, accidental contaminations). We use SATURATE in order to optimize the modified objective function $\overline{F}'_c$ described in Section 6.5.1, for increasing numbers of sensors k. We also use the greedy algorithm to optimize sensor placements, ignoring possible sensor failures. For both algorithms, we compute the expected scores (penalty reductions Z_1 and Z_2) in the case of no



(a) *[W] Sensor failures for Z_1*

(b) *[W] Sensor failures for Z_2*

Figure 12: (a,b) compare Greedy (ignoring sensor failures) and SATURATE (optimizing for the worst-case sensor failure) on water network data with detection time (a) and population affected (b) scores.

sensor failure, and in the case of a single, worst-case sensor failure. Figure 12(a) presents the results for the time to detection objective (Z_1). We can see, that initially, with small numbers of sensors, failures can strongly diminish the Z_1 score. However, as the number of sensors increases, the placement scores optimized using SATURATE for sensor failures quickly approach those of Greedy in the case of no sensor failures. Hence, even if only a small number of sensors are placed, SATURATE can quickly exploit redundancy and find sensor placements, which perform well both with and without sensor failures. On the other hand, when not taking sensor failures into account, such failures can drastically diminish the utility of a placed set of sensors. Figure 12(b) presents analogous results when minimizing the affected population (Z_2).

8.5 Parameter Uncertainty

We also conducted experiments on selecting variables under parameter uncertainty (cf., Section 6.2). More specifically, we consider a sensor placement problem for monitoring temperature in a building. In such a problem, we would like to place sensors in order to get accurate predictions at various times of the day. However, since phenomena such as temperature in buildings change over time, at different times of the day, different placements would be most informative.

In our experiment, we consider the temperature data set [T], and learn four models, described by parameters $\theta_1, \ldots, \theta_4$, during four six-hour time periods over the day: 12am-6am, 6am-12pm, 12pm-6pm and 6pm-12am. As models, we use the empirical covariances $\Sigma^{(\theta_i)}$ from the corresponding time periods of the 5 day historical training data. We also use the single model Σ for the entire day, as described in Section 8.1. We then use the greedy algorithm to optimize sensor placement of increasing sizes for the single model Σ, optimizing the average variance reduction objective function. Similarly, we use SATURATE to optimize the minimum variance reduction over the four models $\Sigma^{(i)}$, normalized by the average variance over the entire space.

Subsequently, we used both placements to compute the average Root Mean Squared (RMS) prediction error over the entire day on 2 days of held out test data. We also computed the maximum

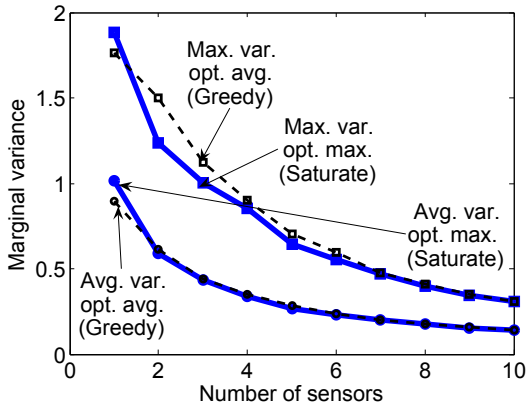


Figure 13: [T] Average and Maximum variance when optimizing for four different covariance models obtained during different parts of the day.

RMS error over the four six-hour time periods. Figure 13 presents the results of this experiment. While the average RMS error is roughly equal for both placements, the maximum RMS error is larger for the greedy sensor placement, as compared to the robust placement of SATURATE, especially for small numbers of sensors (six and less sensors).

9. Reducing the Number of Objective Functions

In many of the examples considered in Section 6, the number m of objective functions F_i can be quite large (e.g., one F_i per parameter setting, or outbreak scenario), which impacts both the running time (which depends linearly on m) and the approximation guarantees (which depend logarithmically on m) of SATURATE. Hence, showing that we can work with a smaller set of objectives has both computational and theoretical advantages.

9.1 Removal of Dominated Strategies

One direct approach to eliminate objective functions (and hence speed up computation and improve the approximation guarantee) is to remove dominated objectives. An objective function F_i is dominated by another objective F_j, if $F_i(A) \geq F_j(A)$ for all sets $A \subseteq V$. Hence, an F_i is dominated by F_j if an adversary can always reduce our score by choosing F_j instead of F_i. For example, when considering sensor failures or feature deletion (as discussed in Section 6.5), for two sets $B \subseteq B'$, the objective F_B is dominated by the objective $F_{B'}$, that is, the score decreases more if more sensors fail. Similarly, in the case of outbreak detection, some outbreak scenarios have much more impact on the network than others. Even though objective functions measuring the impact reductions F_i for scenarios $i \in I$ might not be *exactly* dominated, they might be ε-dominated, that is, $F_i(A) \geq F_j(A) - \varepsilon$ for some $\varepsilon > 0$ and all $A \subseteq V$. In such cases, these approximately dominated scenarios can be removed, incurring at most an error of ε in the quality of the approximate solution.

9.2 Constraint Generation

Another possible approach to reduce the number m of objective functions is constraint generation (cf., Benders 1962). In this approach, one starts with an arbitrary single objective function, F_1. In iteration $j+1$, ($j \geq 1$), after functions $F_1, \ldots, F_j$ have been considered, one searches for set A_j maximizing $\max_A \min_{1 \leq i \leq j} F_j(A)$. Subsequently, one selects F_{j+1} minimizing $\min_i F_i(A_j)$. The iteration terminates once F_{j+1} is contained in the already selected objectives $F_1, \ldots, F_j$. Another option is to terminate once the new objective F_{j+1} is ε-dominated by some objective F_i, $1 \leq i \leq j$. In this case, the approximate solution is guaranteed to incur at most an absolute error of ε as compared to the optimal solution.

In order to implement this constraint generation scheme, one must be able to efficiently solve problem $\min_i F_i(A_j)$. In some settings, this problem might admit an efficient (perhaps approximate) solution. In many problems, such as the experimental design setting, one actually wants to perform well against an (uncountably) infinite set of possible objective functions, corresponding to parameters $\theta \in D$ in some (typically compact and convex set D). In such a setting, $\min_\theta F_\theta(A_j)$ could potentially be (at least heuristically) solved using a numerical optimization approach such as a conjugate gradient method.

10. Related Work

In this section, we review related work in submodular function optimization, robust discrete optimization, robust methods in statistics, sensor placement, game theory and machine learning.

10.1 Submodular Function Optimization

In their seminal work, Nemhauser et al. (1978) and Wolsey (1982) analyze the greedy algorithm for optimizing monotonic submodular functions. Lovász (1983) discusses the relationship between submodular functions and convexity. He also shows that under certain conditions, the minimum of two submodular functions remains submodular (and hence can be efficiently optimized using the greedy algorithm). The objective functions resulting from observation selection problems typically do not satisfy these properties, and, as we have shown, the greedy algorithm can perform arbitrarily badly. Fujito (2000) uses submodularity of truncated functions to find sets with partial submodular coverage; however, they do not consider the case of multiple objectives, which we address in this paper. Bar-Ilan et al. (2001) consider covering problems for a generalization of submodular functions; they use a similar binary search technique combined with multiple applications of the greedy algorithm. Their approach does not apply to maximizing the minimum over a set of submodular functions. Golovin and Streeter (2008) present an algorithm for online maximization of a single submodular set function. An interesting question for future work would be to investigate whether our approach for maximizing the minimum over a collection of submodular functions can be generalized to an online setting as well.

A large part of the theory of optimizing submodular functions is concerned with *minimizing* instead of maximizing a single submodular function. Queyranne (1995) present the first algorithm for minimizing symmetric submodular functions; Iwata et al. (2001) and Schrijver (2000) present combinatorial algorithms for minimizing *arbitrary* (not necessarily symmetric) submodular functions.

10.2 Robust Discrete Optimization

Robust optimization of submodular functions is an instance of a robust discrete optimization problem. In such problems, the goal generally is to perform well with respect to a worst-case choice of evaluation scenario. Other instances of robust discrete problems have been studied by a number of authors. Kouvelis and Yu (1997) introduce several notions of robust discrete problems, presents hardness results and a class of robust problems that can be optimally solved. Averbakh (2001) shows that a class of robust optimization problems (selecting a k-element subset of elements of minimum cost) is solvable in polynomial time if the uncertain cost coefficients are contained in an interval, but NP-hard under an arbitrary (finite) set of adversarially chosen scenarios. Bertsimas and Sim (2003) proposes a class of robust mixed integer programs, accommodating uncertainty both in cost and data coefficients. They show that in certain cases (robust matching, spanning tree, etc.), the robust formulations are solvable in polynomial time if the non-robust problem instances are solvable in polynomial time. In the case of NP-hard but α-approximable non-robust problems, they show that the corresponding robust formulations also remain α-approximable. However, their results do not transfer to our setting of robust submodular optimization, since in this case, even though non-robust solutions are $(1-1/e)$ approximable, the non-robust formulation does not admit any approximation guarantees (cf., Section 3).

10.3 Robust Methods in Statistics

In this section, we review related work in robust experimental design and robust spatial prediction.

10.3.1 ROBUST EXPERIMENTAL DESIGN

Experimental design under parameter uncertainty has been studied in statistics; most of the earlier work is reviewed in the excellent survey of Chaloner and Verdinelli (1995). In the survey, the authors discuss Bayesian approaches to handling parameter uncertainty, as well as robust Bayesian (cf., Berger 1984) approaches, which perform worst-case analyses over prior and likelihood functions. In experimental design, most approaches have focused on *locally* optimal designs, that is, those selecting an optimal design based on a linearization around an initial parameter estimate, for reasons of computational tractability. In order to cope with uncertainty in the initial parameter estimates around which linearization is performed, heuristic techniques have been developed, such as the SDP based approach of Flaherty et al. (2006), or a clustering heuristic described by Dror and Steinberg (2006). We are not aware of approaches which allow to find designs in the context of such parameter uncertainty that bear theoretical guarantees similar to the approach described in this paper.

10.3.2 MINIMAX KRIGING

Minimizing the maximum predictive variance in Gaussian Process regression has been proposed as a design criterion by Burgess et al. (1981) and since then extensively used. (cf., Sacks and Schiller, 1988; van Groenigen and Stein, 1998). To our knowledge, prior to this work, no algorithms with approximation guarantees are known for this criterion.

Several authors consider the problem of spatial prediction under unknown covariance parameters. Pilz et al. (1996) describes an approach for selecting—for a fixed set of observed sites—the Kriging estimate minimizing the maximum prediction error, where the worst-case over a fixed class of covariance functions is assumed. Wiens (2005) consider a similar setting but also addresses

the design problem of choosing locations in order to minimize the mean squared prediction error against the worst-case covariance function. Algorithmically, Wiens (2005) use the simulated annealing algorithm described in Section 8.1 with 7 tuned parameter settings. Note that the SATURATE algorithm can be used in this context as well.

10.4 Sensor Placement and Facility Location

Carr et al. (2006) consider the problem of robust sensor placements in water distribution networks. They formulate Mixed Integer Programs for selecting sensor placements robust against uncertainty in adversarial strategies and in water demands. Due to computational complexity of Mixed Integer Programming, in their experiments, they used only small networks of at most 470 nodes. SATURATE can potentially be applied to handle uncertainty in demands as well, which is an interesting direction for future work. Watson et al. (2006) consider different notions of robustness in the context of water distribution networks, intended to remove some of the pessimistic assumptions of purely robust sensor placements. They develop integer programs, as well as heuristics, and apply them to networks of similar size as the one considered in this paper. Their local search heuristic performs a sequence of local moves similar to those performed by the simulated annealing algorithm considered in Section 8.3, and does not provide any theoretical guarantees.

Closely related to the adversarial outbreak detection problem is the *k-center* problem. In this problem, one is given a graph $G = (V, E)$ along with a distance function defined over pairs of nodes in V. The goal is to select a subset $A \subseteq V$ of size at most k, such that the maximum distance between any unselected node $s \in V \setminus A$ and its nearest center $s' \in A$ is minimized. For this problem, Minieka (1970) discuss a technique reducing the solution of this problem to a sequence of set cover problems combined in a binary search, similar in spirit to SATURATE. However, they do not discuss any implications regarding approximation guarantees, and do not consider the case of arbitrary submodular functions. Mladenovic et al. (2003) presents a Tabu search heuristic for k-center, also without theoretical guarantees. Gonzalez (1985) and Hochbaum and Shmoys (1985) present a 2 approximation for the k-center problem in the case of symmetric distance functions satisfying the triangle inequality. Panigrahy and Vishwanathan (1998) present a $\log^*(n)$ approximation in the case of distance functions satisfying the asymmetric triangle inequality, which is shown to be best possible by Chuzhoy et al. (2005). Chuzhoy et al. (2005) also show that even for bicriterion algorithms (such as SATURATE), k-center is $\log^*(n)$ hard to approximate, even if $O(k)$ additional centers can be selected. Note that SATURATE can be used to solve k-center problems (without any requirements on symmetry or on the triangle inequality), hence the bicriterion hardness result of Chuzhoy et al. (2005) gives further evidence on the tightness of the guarantees described in Section 5.

Anthony et al. (2008) consider robust and stochastic notions of facility location problems (such as k-center and k-median, where, instead of the maximum distance the average distance is optimized). In contrast to the robust problems in this paper which want to select k elements to *maximize* the minimum value achieved by these k elements over the m scenarios, the problems in Anthony et al. (2008) try to select k "centers" in a metric space to *minimize* the maximum cost incurred over the m scenarios—where the cost is some function of the distances between non-selected vertices to the selected centers. For several such robust cost-minimization problems in cases where distances satisfy the symmetric triangle inequality, they present an algorithm that opens k "centers" and achieves an approximation ratio of $O(\log n + \log m)$ (where n is the number of nodes in the

graph, and m is the number of scenarios): this should be compared to the impossibility results for approximating robust value-maximization problems presented in this paper.

10.5 Relationship to Game Theory and Allocation Problems

The RSOS problem can be viewed as the problem of finding an optimal pure strategy for a zero-sum matrix game with player ordering. In this matrix game, the rows would correspond to the possible sensor placements, and the columns would correspond to the objective functions F_i. The entry for cell (A, F_i) is our payoff $F_i(A)$. In the RSOS problem, we want to select a row of the matrix, our adversary selects a column F_i (knowing our choice A, hence the player ordering) minimizing our score $F_i(A)$. A very related class of game theoretic problems are *allocation problems*. In these problems, one is typically given a set V of objects, and the goal is to allocate the objects to m agents (bidders), each of whom has a (potentially different) valuation function $F_i(A_i)$ defined over subsets of received items A_i. The problem of finding the best such allocation (partition) is NP-hard, but recently, several approximation algorithms have been proposed. The allocation problem most similar to the RSOS problem is

$$\pi^* = \operatorname*{argmax}_{\text{partition } \pi=(A_1,\ldots,A_m)} \min_i F_i(A_i).$$

The main difference is that in the allocation problem, the full set V is partitioned into subsets $A_1,\ldots,A_m$, and the functions F_i are evaluated on the respective subset A_i each. In the case of *additive* objective functions F_i, Asadpour and Saberi (2007) provide an $O(\sqrt{k}\log^3 k)$ approximation algorithm. In the case of the function being *subadditive* (which is implied by, and is more general than, submodularity), Ponnuswami and Khot (2007) present an $O(2k-1)$ approximation algorithm. For settings where the *sum* of the valuations is optimized, that is,

$$\pi^* = \operatorname*{argmax}_{\text{partition } \pi=(A_1,\ldots,A_m)} \sum_i F_i(A_i),$$

Feige (2006) develop a randomized 2-approximation for subadditive and $1-1/e$ approximation for submodular valuation functions.

The problem of trading off safety (i.e., improvements in worst-case scores) and average case performance has been studied by several authors. Johanson et al. (2007) consider the problem of opponent modeling in games, and develop an algorithm which can exploit opponents which it can accurately model, and falls back to a safe (Nash) strategy in case the models do not capture the opponents behavior. Their algorithm has a tradeoff parameter which controls the eagerness of exploiting, and they present Pareto-curves similar to those presented in Section 7.5. However, their approach does not apply to our robust submodular observation selection setting. Watson et al. (2006) consider different optimization problem formulations allowing to control risk in the water distribution network monitoring application, but they only present heuristic algorithms without guarantees for coping with large networks.

10.6 Relationship to Machine Learning

Submodular function optimization has found increasing use in machine learning. The algorithm of Queyranne (1995) for minimizing symmetric submodular functions has been used for learning graphical models by Narasimhan and Bilmes (2004) and for clustering by Narasimhan et al. (2005).

We are not aware of any work on optimizing the minimum over a collection of submodular functions.

Observation selection approaches have been used in the context of active learning (cf., Sollich, 1996; Freund et al., 1997; Axelrod et al., 2001; MacKay, 1992; Cohn, 1994). Test point selection has been used to minimize average predictive variance in Gaussian Processes regression by Seo et al. (2000), and to speed up Gaussian Process inference in the Informative Vector Machine (IVM) by Seeger et al. (2003); Lawrence et al. (2003). In these approaches, the sequential setting is considered, where previous measurements are taken into account when deciding on the next observation to make. A note by Seeger (2004) proves that the greedy algorithm in the IVM optimizes a submodular function. The extension of the robust techniques discussed in this paper, which address the a priori selection problem (i.e., observations are selected before measurements are obtained), to the sequential setting is an important direction for future research.

Balcan et al. (2006) consider the problem of active learning in the presence of *adversarial* noise. While their method is very different, our results potentially generalize to active learning settings, since, as Hoi et al. (2006) show, certain active learning objectives are (approximately) submodular.

Price and Messinger (2005) consider the problem of constructing recommendation sets, and show that this problem is an instance of a k-median problem (cf., Section 10.4). The analogue of the k-center problem in the preference set construction would be to construct a preference set which maximizes the utility of displayed items under worst-case instantiation of the parameters. This analogue seems natural, and an interesting direction for future work would be to explore the use of SATURATE in the recommendation set context.

10.7 Relationship to Previous Work of the Authors

A previous version of this paper appeared in (Krause et al., 2007a). The present version is significantly extended, providing new theoretical analyses (described in Section 7, Section 9), new examples demonstrating the generality of the observation selection problem (Section 6) and additional empirical results (Section 8). In previous work, the authors demonstrated that several important observation selection objectives are submodular (Krause et al., 2007b; Leskovec et al., 2007; Krause and Guestrin, 2005, 2007a). Krause et al. (2006) consider the problem of optimizing the placement of a network of wireless sensors. In this context, the chosen locations must be both informative and communicate well, constraining the chosen locations not to be too far apart. Singh et al. (2007) and Meliou et al. (2007) consider the problem of planning informative paths for multiple robots, where the informativeness is modeled using a submodular objective function, and a constraint on path lengths connecting the locations is specified. In the context of such more complex (communication and path) constraints—similarly to the robust setting—the greedy algorithm can fail arbitrarily badly, and more complex algorithms have to be developed. Using the techniques described in Section 7.4, both approaches can be made robust with respect to a worst-case submodular function.

11. Conclusions

In this paper, we considered the RSOS problem of robustly selecting observations which are informative with respect to a worst-case submodular objective function. We demonstrated the generality of this problem, and showed how it encompasses the problem of sensor placements which minimize the maximum posterior variance in Gaussian Process regression, variable selection under parameter uncertainty, robust experimental design, and detecting events spreading over graphs, even in the

case of adversarial sensor failures. In each of these settings, the individual objectives are submodular and can be approximated well using, for example, the greedy algorithm; the robust objective, however, is not submodular.

We proved that there cannot exist any approximation algorithm for the robust optimization problem if the constraint on the observation set size must be exactly met, unless P = NP. Consequently, we presented an efficient approximation algorithm, SATURATE, which finds observation sets which are guaranteed to be least as informative as the optimal solution, and only logarithmically more expensive. In a strong sense, this guarantee is the best possible under reasonable complexity theoretic assumptions.

We provided several extensions to our methodology, accommodating more complex cost functions (non-uniform observation costs, communication and path costs). Additionally, we described how a compromise between worst-case and average-case performance can be achieved. We also discussed several approaches for reducing the number of objective functions, improving both running times and theoretical guarantees.

We extensively evaluated our algorithm on several real-world problems. For Gaussian Process regression, for example, we showed that SATURATE compares favorably to state-of-the-art heuristics, while being simpler, faster, and providing theoretical guarantees. For robust experimental design, SATURATE performs favorably compared to SDP based approaches. We believe that the ideas developed in this paper will help the development of robust monitoring systems and provide new insights for adapting machine learning algorithms to cope with adversarial environments.

Acknowledgments

We would like to thank Michael Bowling for helpful discussions. We would also like to thank the anonymous referees for their helpful insights and detailed feedback. This work was partially supported by NSF Grants No. CNS-0509383, CNS-0625518, CCF-0448095, CCF-0729022, ARO MURI W911NF0710287 and a gift from Intel. Anupam Gupta and Carlos Guestrin were partly supported by Alfred P. Sloan Fellowships, Carlos Guestrin by an IBM Faculty Fellowship and an ONR Young Investigator Award N00014-08-1-0752 (2008-2011). Andreas Krause was partly supported by a Microsoft Research Graduate Fellowship.

Appendix A. Proofs

Proof [Theorem 3] Consider a hitting set instance with m subsets $S_i \subseteq V$ on a ground set V. Our task is to select a set $A \subseteq V$ with which intersects all sets S_i, and such that $|A| = k$ is as small as possible. For each set S_i, define a function F_i such that $F_i(A) = 1$ if A intersects S_i, and 0 otherwise. It can be seen that F_i is clearly monotonic. F_i is also submodular, since for $A \subseteq B \subseteq V$ and $x \in V \setminus B$, if $F_i(B) = 0$ and $F_i(B \cup \{x\}) = 1$, then it $x \in S_i$, hence $F_i(A \cup \{x\}) = 1$ and $F_i(A) = 0$. Now assume the optimal hitting set A^* is of size k. Hence $\min_i F_i(A^*) = 1$. If there were an algorithm for solving Problem (2) with approximation guarantee $\gamma(n)$ it would select a set A' of size $|A'| \leq k$ with $\min_i F_i(A') \geq \gamma(n) \min_i F_i(A^*) = \gamma(n) > 0$. But $\min_i F_i(A') > 0$ implies $\min_i F_i(A') = 1$, hence A' would be a hitting set. Hence, this approximation algorithm would be able to decide, whether there exists a hitting set of size k, contradicting the NP-hardness of the hitting set problem (Feige, 1998). ■

Proof [Lemma 4] Wolsey (1982) proves that, given a monotonic submodular function F on a ground set V, it holds that that greedy algorithm (GPC), applied to the optimization problem

$$\min_A |A| \text{ such that } F(A) = F(V)$$

returns a solution A' such that $|A'| \leq |A^*|(1+\log\max_{s\in V} F(\{s\}))$, where A^* is an optimal solution. We apply Wolsey's result to the monotonic submodular function $\overline{F}_c$. In order to use GPC in the inner loop of the binary search over c, we need to make sure that the approximation guarantee for the greedy algorithm is independent of c. This can be achieved by choosing

$$\alpha = 1 + \log\left(\max_{s\in V}\sum_i F_i(\{s\})\right) \geq 1 + \log\left(m\max_{s\in V}\overline{F}_c(\{s\})\right),$$

that is, the choice of α stated in Lemma 4 is independent of the truncation threshold c. ■

Proof [Theorem 5] Lemma 4 proves that during each of the iterations of the saturation algorithm it holds that $\min_i F_i(A^*) \leq c_{\max}$, where A^* is an optimal solution. Furthermore, it holds that $\min_i F_i(A_{best}) \geq c_{\min}$, and $|A_{best}| \leq \alpha k$. Since the F_i are integral, if $c_{\max} - c_{\min} < \frac{1}{m}$ then it must hold that $\min_i F_i(A_{best}) \geq \min_i F_i(A^*)$ as claimed by Theorem 5.

For the running time, since at the first iteration, $c_{\max} - c_{\min} \leq \min_i F_i(V)$, and $c_{\max} - c_{\min}$ is halved during each iteration, it follows that after $1 + \lceil \log_2 m \min_i F_i(V) \rceil$ iterations, $c_{\max} - c_{\min} < \frac{1}{m}$, at which point the algorithm terminates. During each iteration, Algorithm 1 is invoked once, which requires $O(|V|^2 m)$ function evaluations. ■

Proof [Theorem 6] We use the same hitting set construction as in Theorem 3. If there were an algorithm for selecting a set A' of size $|A'| \leq \beta k$ with $\min_i F_i(A') = 1$, and $\beta \leq (1-\varepsilon)\alpha$, for some fixed $\varepsilon > 0$, then we would have an approximation algorithm for hitting set with guarantee $(1-\varepsilon)\log m$ which would imply NP $\subseteq$ DTIME$(n^{\log\log n})$ (Feige, 1998). ■

Proof [Theorem 8] The proof is analogous to the proof of Theorem 5. The approximation guarantee α is established by noticing that the greedy algorithm is applied to the modified (integral) objective

$$\overline{F}_{c_{wc},c_{ac}}(A) = \sum_i \min\{F_i(A), c_{wc}\} + \min\left\{\sum_i F_i(A), mc_{ac}\right\}.$$

The guarantee α is obtained from the analysis of the greedy submodular coverage algorithm of Wolsey (1982), similar to Lemma 4. Approximate Pareto-optimality follows directly from Pareto-optimality of any solution to (10). ■

References

B. M. Anthony, V. Goyal, A. Gupta, and V. Nagarajan. A plant location guide for the unsure. In *SODA*, 2008.

A. Asadpour and A. Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC*, pages 114–121, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-631-8. doi: http://doi.acm.org/10.1145/1250790.1250808.

I. Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90:263–272, 2001.

S. Axelrod, S. Fine, R. Gilad-Bachrach, R. Mendelson, and N. Tishby. The information of observations and application for active learning with uncertainty. Technical report, Jerusalem: Leibniz Center, Hebrew University, 2001.

M. F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *ICML*, 2006.

J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized submodular cover problems and applications. *Theoretical Computer Science*, 250(1-2):179–200, January 2001.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

J. Berger. *Robustness of Bayesian Analyses*, chapter The robust Bayesian viewpoint, page 63144. North-Holland, 1984.

D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98:49–71, 2003.

A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. In *FOCS*, page 46, 2003.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge UP, March 2004.

T.M. Burgess, R. Webster, and A.B. McBratney. Optimal interpolation and isarithmic mapping of soil properties. iv. sampling strategy. *Journal of Soil Science*, 32:643–659, 1981.

R. D. Carr, H. J. Greenberg, W. E. Hart, G. Konjevod, E. Lauer, H. Lin, T. Morrison, and C. A. Phillips. Robust optimization of contaminant sensor placement for community water systems. *Mathematical Programming Series B*, 107:337–356, 2006.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3): 273–304, Aug. 1995. ISSN 08834237.

J. Chuzhoy, S. Guha, E. Halperin, S. Khanna, G. Kortsarz, R. Krauthgamer, and J. Naor. Asymmetric k-center is $\log^* n$-hard to approximate. *Journal of the ACM*, 52(4):538–551, 2005.

D. A. Cohn. Neural network exploration using optimal experiment design. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 679–686. Morgan Kaufmann Publishers, Inc., 1994.

N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991.

A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *ACM Symposium on the Theory of Computing (STOC)*, 2008.

H. A. Dror and D. M. Steinberg. Robust experimental design for multivariate generalized linear models. *Technometrics*, 48(4):520–529, 2006.

U. Feige. On maximizing welfare when utility functions are subadditive. In *STOC*, 2006.

U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4), 1998.

P. Flaherty, M. Jordan, and A. Arkin. Robust design of biological experiments. In *NIPS*, 2006.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

T. Fujito. Approximation algorithms for submodular set cover with applications. *TIEICE*, 2000. URL `citeseer.ist.psu.edu/article/fujito00approximation.html`.

A. Globerson and S. Roweis. Nightmare at test time: Robust learning by feature deletion. In *ICML*, 2006.

D. Golovin and M. Streeter. Online algorithms for maximizing submodular set functions. In *Submitted to SODA*, 2008.

T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38(2-3):293–306, 1985. ISSN 0304-3975.

C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in Gaussian processes. In *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML)*, 2005.

T. C. Harmon, R. F. Ambrose, R. M. Gilbert, J. C. Fisher, M. Stealey, and W. J. Kaiser. High resolution river hydraulic and water quality characterization using rapidly deployable networked infomechanical systems (nims rd). Technical Report 60, CENS, 2006.

D. Hochbaum and D. Shmoys. A best possible heuristic for the *k*-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, 2006.

S. Iwata, L. Fleischer, and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.

M. Johanson, M. Zinkevich, and M. Bowling. Computing robust counter-strategies. In *NIPS*, 2007.

D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting steiner tree problem: theory and practice. In *SODA*, 2000.

P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers, 1997.

A. Krause and C. Guestrin. Near-optimal value of information in graphical models. In *UAI*, 2005.

A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *AAAI Nectar track*, 2007a.

A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: An exploration—exploitation approach. In *ICML*, 2007b.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2006.

A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Selecting observations against adversarial objectives. In *NIPS*, 2007a.

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *To appear in the JMLR*, 2007b.

G. Laporte and S. Martello. The selective travelling salesman problem. *Disc. App. Math*, 26:193–207, 1990.

N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems (NIPS) 16*, 2003.

J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.

L. Lovász. Submodular functions and convexity. *Mathematical Programming - State of the Art*, pages 235–257, 1983.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, 1990.

A. Meliou, A. Krause, C. Guestrin, and J. M. Hellerstein. Nonmyopic informative path planning in spatio-temporal models. In *AAAI*, 2007.

E. Minieka. The *m*-center problem. *SIAM Rev*, 12(1):138–139, 1970.

N. Mladenovic, M. Labbé, and P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003.

M. Narasimhan and J. Bilmes. Pac-learning bounded tree-width graphical models. In *Uncertainty in Artificial Intelligence*, 2004.

M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. In *NIPS*, 2005.

G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

A. Ostfeld, J. G. Uber, and E. Salomons. Battle of water sensor networks: A design challenge for engineers and algorithms. In *8th Symposium on Water Distribution Systems Analysis*, 2006.

A. Ostfeld, J. G. Uber, E. Salomons, J. W. Berry, W. E. Hart, C. A. Phillips, J. Watson, G. Dorini, P. Jonkergouw, Z. Kapelan, F. di Pierro, S. Khu, D. Savic, D. Eliades, M. Polycarpou, S. R. Ghimire, B. D. Barkdoll, R. Gueli, J. J. Huang, E. A. McBean, W. James, A. Krause, J. Leskovec, S. Isovitsch, J. Xu, C. Guestrin, J. VanBriesen, M. Small, P. Fischbeck, A. Preis, M. Propato, O. Piller, G. B. Trachtman, Z. Y. Wu, and T. Walski. The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *To appear in the Journal of Water Resources Planning and Management*, 2008.

R. Panigrahy and S. Vishwanathan. An $O(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *Journal of Algorithms*, 27(2):259–268, 1998.

C. H. Papadimitriou and M. Yannakakis. The complexity of tradeoffs, and optimal access of web sources. In *FOCS*, 2000.

J. Pilz, G. Spoeck, and M. G. Schimek. *Geostatistics Wollongong*, volume 1, chapter Taking account of uncertainty in spatial covariance estimation, pages 302–313. Kluwer, 1996.

A. K. Ponnuswami and S. Khot. Approximation algorithms for the max-min allocation problem. In *APPROX*, 2007.

R. Price and P. R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *AAAI*, 2005.

M. Queyranne. A combinatorial algorithm for minimizing symmetric submodular functions. In *SODA*, 1995.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Process for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

T. G. Robertazzi and S. C. Schwartz. An accelerated sequential algorithm for producing D-optimal designs. *SIAM Journal of Scientific and Statistical Computing*, 10(2):341–358, March 1989.

L. A. Rossman. The epanet programmer's toolkit for analysis of water distribution systems. In *Annual Water Resources Planning and Management Conference*, 1999.

J. Sacks and S. Schiller. *Statistical Decision Theory and Related Topics IV, Vol. 2*. Springer, 1988.

A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B*, 80(2):346–355, 2000. ISSN 0095-8956.

M. Seeger. Greedy forward selection in the informative vector machine. Technical report, University of California at Berkeley, 2004.

M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2003.

S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 241–246, 2000.

A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin. Efficient planning of informative paths for multiple robots. In *IJCAI*, 2007.

P. Sollich. Learning from minimum entropy queries in a large committee machine. *Physical Review E*, 53:R2060–R2063, 1996.

J.W. van Groenigen and A. Stein. Constrained optimization of spatial sampling using continuous simulated annealing. *J. Environ. Qual.*, 27:1078–1086, 1998.

V. V. Vazirani. *Approximation Algorithms*. Springer, 2003.

J. Watson, W. E. Hart, and R. Murray. Formulation and optimization of robust sensor placement problems for contaminant warning systems. In *Water Distribution System Symposium*, 2006.

M. Widmann and C. S. Bretherton. 50 km resolution daily precipitation for the pacific northwest. http://www.jisao.washington.edu/data_sets/widmann/, May 1999.

D. P. Wiens. Robustness in spatial studies ii: minimax design. *Environmetrics*, 16:205–217, 2005.

L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982.

Journal of Machine Learning Research 9 (2008) 2803-2846 Submitted 6/08; Published 12/08

Magic Moments for Structured Output Prediction

Elisa Ricci ELISA.RICCI@DIEI.UNIPG.IT
Dept. of Electronic and Information Engineering
University of Perugia
06125 Perugia, Italy

Tijl De Bie TIJL.DEBIE@GMAIL.COM
Dept. of Engineering Mathematics
University of Bristol
Bristol, BS8 1TR, UK

Nello Cristianini NELLO@SUPPORT-VECTOR.NET
Dept. of Engineering Mathematics and Dept. of Computer Science
University of Bristol
Bristol, BS8 1TR, UK

Editor: Michael Collins

Abstract

Most approaches to structured output prediction rely on a hypothesis space of *prediction functions* that compute their output by maximizing a linear *scoring function*. In this paper we present two novel learning algorithms for this hypothesis class, and a statistical analysis of their performance. The methods rely on efficiently computing the first two moments of the scoring function over the output space, and using them to create convex objective functions for training. We report extensive experimental results for sequence alignment, named entity recognition, and RNA secondary structure prediction.

Keywords: structured output prediction, discriminative learning, Z-score, discriminant analysis, PAC bound

1. Introduction

The last few years have seen a growing interest in learning algorithms that operate over structured data: given a set of training input-output pairs, they learn to predict the output corresponding to a previously unseen input, where either the input or the output (or both) are more complex than traditional data types such as vectors.

Examples of such problems abound: learning to align biological sequences, learning to parse strings, learning to translate natural language, learning to find the optimal route in a graph, learning to understand speech, and much more. This problem setting subsumes as a special case the standard regression, binary classification, and multiclass classification problems. In fact in many cases the structured output prediction approach matches practice more closely. However, this broad generality and applicability comes with a number of significant theoretical and practical challenges.

In standard regression the output space is real-valued, and in classification the output space consists of a relatively small unstructured set of labels. In contrast, in structured output prediction the output space is typically massive, containing a rich structure relating the different output values

with each other. Because of this, even the prediction task itself requires a search (or optimization) over the complete output space, which in itself is often nontrivial. A fortiori, the task of learning to predict poses important new challenges in comparison with standard machine learning approaches such as regression and classification.

1.1 Graphical and Grammatical Models for Structured Data

An immediate approach for structured output prediction would be to use a probabilistic model jointly over the input and the output variables. Probabilistic graphical models (PGMs) or stochastic context free grammars (SCFGs) are two examples of techniques that allow one to specify probabilistic models for a variety of inputs and outputs, explicitly encoding the structure that is present. For a given input, the predicted output can then be found as the one that maximizes the a posteriori probability. This way of predicting structured outputs is referred to as maximum a posteriori (MAP) estimation. The learning phase then boils down to modeling the distribution of the joint of input and output data.

However, it is well known that this indirect approach of first modeling the distribution (disregarding the prediction task of interest) and subsequently using MAP estimation for prediction, risks to be suboptimal. Instead a direct discriminative approach is more appropriate, which directly focuses on the prediction task of interest. Such methods, known as discriminative learning algorithms (DLAs), make predictions by optimizing a scoring function over the output space, where this scoring function has not necessarily a probabilistic interpretation.

Recently studied DLAs include maximum entropy Markov models (McCallum et al., 2000), conditional random fields (CRFs) (Lafferty et al., 2001), re-ranking with perceptron (Collins, 2002b), hidden Markov perceptron (HMP) (Collins, 2002a), sequence labeling with boosting (Altun et al., 2003a), maximal margin (MM) algorithms (Altun et al., 2003b; Taskar et al., 2003; Tsochantaridis et al., 2005), Gaussian process models (Altun et al., 2004), and kernel conditional random fields (Lafferty et al., 2004).

Interestingly, both the generative modeling approach and the DLAs mentioned above make use of formally the same hypothesis class of prediction functions. In particular, they all make use of a scoring function that is linear in a set of parameters to score each element of the output space. In the generative approach, this linear function is the log-probability of the joint of the input and output data; in the discriminative approach this can be any linear function. The actual prediction function then selects the output that achieves the highest value of the scoring function (i.e., the highest score). In the generative approach this means that the a posteriori (log)-probability of the output is maximized, such that the MAP estimate is obtained as pointed out above.

1.2 The Contributions of this Paper

In this paper we will adopt the hypothesis space of prediction functions defined as above. The distribution of scores induced by any hypothesis over all possible outputs is a central concept in various approaches, and can be used to compare hypotheses, and hence to train. For example MM approaches (Altun et al., 2003b; Taskar et al., 2003; Tsochantaridis et al., 2005) prescribe to seek hypotheses that make the score of the correct outputs in the training set larger than all incorrect ones (by a certain margin).

We argue that the problem can be better approached by considering the entire distribution of the scores over the output space, and in particular by computing its first two moments. Different

choices of parameters can be assessed by comparing (a function of) those moments. Such an approach would account for all possible output values at once, rather than just the ones with a high score as in the maximum margin approaches. However these moments cannot be computed by brute force enumeration: in all practical cases the output space is far too large to exhaustively traverse it. Nevertheless in this paper we show how the first and second order moments can often be computed efficiently by means of dynamic programming (DP), without explicitly enumerating the output space. We provide specific examples of how these moments can be computed for three types of structured output prediction problems: the sequence alignment problem, sequence labeling, and learning to parse with a context free grammar for RNA secondary structure prediction.

We then present two ways in which these moments can be used to design a convex objective function for a learning algorithm. **The first approach** is the maximization of the Z-score, a common statistical measure of surprise, which is large if the scores of the correct outputs in the training set are significantly different from the scores of all incorrect outputs in the output space. We show that the Z-score is a convex cost function, such that it can be optimized efficiently. **A second approach**—also convex—is reminiscent of Fisher's discriminant analysis (FDA). We call this new algorithm SODA (structured output discriminant analysis) since the optimization criterion is a similar function of the first and second order statistics as in FDA.

We report extensive experimental results for the proposed algorithms applied to three different problem settings: learning to align, sequence labeling, and RNA folding.

Finally we derive learning-theoretic bounds on the performance of these algorithms, showing that the SODA cost function is related to the rank of the correct output among the other outputs and analyzing its statistical stability within the Rademacher framework; additionally, we present a general PAC bound that applies to any algorithm using this hypothesis class.

1.3 Outline of this Paper

The rest of the paper is structured as follows: Section 2 formally introduces the problem of structured output learning and the hypothesis space considered. Section 3 deals with the computation of the first and second order moments of the score distribution through DP. In Section 4 we introduce the two algorithms. In Section 5 we present our experimental results, and in Section 6 we outline learning-theoretical bounds, whose proof is however left for the appendix.

2. Learning to Predict Structured Outputs

We address the general problem of learning a *prediction function* $h : \mathcal{X} \to \mathcal{Y}$, with $\mathcal{Y}$ a potentially highly structured space containing a potentially large number N of elements. The learning is based on a training set of input-output pairs $\mathcal{T} = \{(\mathbf{x}^1, \overline{\mathbf{y}}^1), (\mathbf{x}^2, \overline{\mathbf{y}}^2), \ldots, (\mathbf{x}^\ell, \overline{\mathbf{y}}^\ell)\}$ drawn i.i.d. from some fixed but unknown distribution $P(\mathbf{x}, \mathbf{y})$ over $\mathcal{X} \times \mathcal{Y}$. The inputs and the outputs may be highly structured objects that parameterize sequences, trees or graphs. For example in sequence alignment learning the output variables parameterize the alignment between two sequences, in sequence labeling $\mathbf{y}$ is the label sequence associated to the observed sequence $\mathbf{x}$, and when learning to parse $\mathbf{y}$ represents a parse tree corresponding to a given sequence $\mathbf{x}$.

2.1 Scoring Functions, Prediction Functions, and the Hypothesis Space

As in standard machine learning approaches, we consider learning methods that choose the prediction function from a hypothesis space by minimizing a cost function evaluated on the training data. To establish the type of hypothesis space we will consider, we will rely on the notion *scoring function*, which is a function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ that assigns a numerical score $s(\mathbf{x}, \mathbf{y})$ to a pair $(\mathbf{x}, \mathbf{y})$ of input-output variables. Furthermore, we will assume that s is linear in a parameter vector $\theta \in \mathbb{R}^d$:

Definition 1 (Linear scoring function) *A linear scoring function is a function $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ defined as:*

$$s_\theta(\boldsymbol{x}, \boldsymbol{y}) = \theta^T \phi(\boldsymbol{x}, \boldsymbol{y}), \tag{1}$$

where the vector $\phi(\boldsymbol{x}, \boldsymbol{y}) = (\phi_1(\boldsymbol{x}, \boldsymbol{y}),\ \phi_2(\boldsymbol{x}, \boldsymbol{y}),\ \ldots,\ \phi_d(\boldsymbol{x}, \boldsymbol{y}))^T$ is defined by a specified set of integer-valued feature functions $\phi_i : \mathcal{X} \times \mathcal{Y} \to [0, C]$ for a fixed upper bound C.

Based on this, we can define prediction functions as considered in this paper as follows:

Definition 2 (Prediction function) *Given a linear scoring function s_θ, we can define a prediction function $h_\theta : \mathcal{X} \to \mathcal{Y}$ as:*

$$h_\theta(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} s_\theta(\boldsymbol{x}, \boldsymbol{y}). \tag{2}$$

This type of prediction function has been used in previous approaches for structured output prediction. For example, when using a discrete-valued PGM to model the joint distribution of the input and output data, the logarithm of the probability distribution is a linear scoring function as defined above. The vector $\phi(\mathbf{x}, \mathbf{y})$ is then the vector of sufficient statistics, and the parameter vector θ corresponds to the logarithms of the clique potentials or conditional probabilities. Then, a MAP estimator corresponds to a prediction function as defined above. Furthermore, note that each feature function ϕ_i that counts a sufficient statistic is either an indicator function, or the sum of an indicator function evaluated on a set of cliques over which the parameter θ_i is reused. Therefore, each of the features must be an integer between 0 and the number of cliques C, as required for linear scoring functions in Definition 1.

Typically, in PGMs the parameters θ would be inferred by Maximum Likelihood. On the contrary in DLAs θ is computed by minimizing criteria that are more directly linked to the prediction performance. Moreover with DLAs richer feature vectors (with features not necessarily associated to clique potentials or conditional probabilities) are allowed to describe more effectively the relation between input and output variables. This means that the score $s_\theta(\mathbf{x}, \mathbf{y})$ looses its interpretation as a log-likelihood function.

In summary, the hypothesis space we consider in this paper is defined as:

$$\mathcal{H} = \{h_\theta : \theta \in \mathbb{R}^d\}. \tag{3}$$

This is a slightly larger hypothesis space as compared to the one considered in PGMs, since the parameters θ are not restricted to represent log probabilities.

While we choose to abandon the probabilistic interpretations, it is often worthwhile to keep the analogy with PGMs in mind: they teach us when the evaluation of the prediction function (2) can be carried out efficiently by means of Viterbi-like algorithms, despite the huge size of the

output space $\mathcal{Y}$. In fact, it is often convenient to define or derive the scoring function starting from a PGM, to ensure that it is easily maximized by a dynamic programming procedure such as the Viterbi algorithm. Subsequently the constraints on the parameters that are meant to guarantee that the scoring function is a log-probability function can be removed, in order to arrive at a hypothesis space of the form (3).

2.2 Ideal Loss Functions

In order to select an appropriate prediction function from the hypothesis space, a cost function needs to be defined. Here we will provide an overview of a few conceptually interesting cost functions, but which are unfortunately hard to optimize. Nevertheless, they can often be approximated as seen from literature, and as we will demonstrate further on.

Consider a *loss function* $\mathcal{L}_\theta$ that maps the input $\mathbf{x}$ and the true training output $\bar{\mathbf{y}}$ to a positive real number $\mathcal{L}_\theta(\mathbf{x},\bar{\mathbf{y}})$, in some way measuring the discrepancy between the prediction $h_\theta(\mathbf{x})$ and $\bar{\mathbf{y}}$. Empirical risk minimization strategies attempt to find the vector $\theta \in \mathbb{R}^d$ such that the *empirical risk*, defined as:

$$\mathcal{R}_\theta(\mathcal{T}) = \frac{1}{\ell}\sum_{i=1}^{\ell} \mathcal{L}_\theta(\mathbf{x}^i,\bar{\mathbf{y}}^i)$$

is minimized, in hopes that this will guarantee that the expected loss $E\{\mathcal{L}_\theta(\mathbf{x},\bar{\mathbf{y}})\}$ is small as well. Often it is beneficial to introduce regularization in order to prevent overfitting to occur, but let us first consider on the empirical risk itself.

Clearly the choice of the loss function is critical, and different choices may be appropriate in different situations. The simplest one is a natural extension of the zero-one loss in binary classification task, defined as:

$$\mathcal{L}_\theta^{ZO}(\mathbf{x},\bar{\mathbf{y}}) \quad = \quad I(h_\theta(\mathbf{x}) \neq \bar{\mathbf{y}})$$

where $I(\cdot)$ is an indicator function. Unfortunately the zero-one loss function is discontinuous and NP-hard to optimize. Therefore algorithms such as CRFs (Lafferty et al., 2001) and MM methods (Altun et al., 2003b) minimize an upper bound on this loss rather than the loss itself, combined with an appropriate regularization term.

However, in structured output prediction, the zero-one loss is quite crude, in the sense that it makes no distinction in the type of mistake that has been made. For example, assume that the outputs $\mathbf{y}$ are sequences of length m, or vectors: $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. In that case, a wrong prediction is likely to be less damaging if it is due to only one or a few incorrectly predicted symbols in the sequence. A better loss function that distinguishes incorrect predictions in this way is the Hamming loss, originally proposed in Taskar et al. (2003) for MM algorithms:

$$\mathcal{L}_\theta^{H}(\mathbf{x},\bar{\mathbf{y}}) \quad = \quad \sum_j I(h_{\theta,j}(\mathbf{x}) \neq \bar{y}_j),$$

counting the number of elements (i.e., the symbols in a sequence, or coordinates in a vector) of the output where a mistake has been made.

However, the Hamming loss is not necessarily a good measure for the severity of an incorrect prediction. For example, certain sentences can be parsed in totally different ways that can all be correct, with a large Hamming distance separating them. Similarly, RNA molecules can have two

totally different stable fold states, both with functional relevance. Therefore, where perfect prediction of the data cannot be achieved using the hypothesis space considered, it could be more useful to measure the fraction of outputs for which the score is ranked higher than for the correct output. This is the main motivation to use what we call the relative ranking (RR) loss:

$$\mathcal{L}_0^{RR}(\mathbf{x},\overline{\mathbf{y}}) = \frac{1}{N}\sum_{j=1}^{N} I(s(\mathbf{x},\overline{\mathbf{y}}) \leq s(\mathbf{x},\mathbf{y}_j)),$$

We refer to this loss as the relative ranking loss, since the rank divided by the total size of the output space N is computed. This loss and related loss functions have been proposed in Freund et al. (1998), Schapire and Singer (1999) and Altun et al. (2003a).

2.3 Playing with Sequences: Labeling, Aligning and Parsing

In order to further clarify the framework of structured output learning we present three typical problems which we will use in the rest of the paper as illustrative examples: sequence labeling learning, sequence alignment learning and parse learning.

2.3.1 SEQUENCE LABELING LEARNING

In sequence labeling tasks a sequence is taken as an input, and the output to be predicted is a sequence that annotates the input sequence, that is, with a symbol corresponding to each symbol in the input sequence. This problem arises in several application such as gene finding or protein structure prediction in computational biology or named entity recognition and part of speech tagging in the natural language processing field. Traditionally a special type of PGM, namely hidden Markov models (HMMs) (Rabiner, 1989), is used in sequence labeling, where the parameters can be learned by maximum likelihood, and subsequently predictions can be made by MAP estimation. In order to derive a DLA for this setting, we will first derive the prediction function corresponding to MAP estimation based on HMMs, and subsequently remove the constraints on the parameters that allow for the probabilistic interpretation of HMMs. Then an appropriate cost function for discrimination can be optimized to select a good parameter setting.

In an HMM (Fig. 1) there is a sequence of observed variables $\mathbf{x} = (x_1, x_2, ..., x_m) \in \mathcal{X}$ which will be the input in the terminology of the paper, along with a sequence of corresponding hidden variables $\mathbf{y} = (y_1, y_2, ..., y_m) \in \mathcal{Y}$, in the present terminology corresponding to the output sequence to be predicted. Each observed symbol x_i is an element of the observed symbol alphabet Σ_x, and the hidden symbols y_i are elements of Σ_y, with $n_o = |\Sigma_x|$ and $n_h = |\Sigma_y|$ the respective alphabet sizes. Therefore the output space is $\mathcal{Y} = {\Sigma_y}^m$, while $\mathcal{X} = {\Sigma_x}^m$. The number of cliques $C = 2m-1$ of the HMM graphical model is equal to the number of edges.

An HMM is defined as a probabilistic model for the joint distribution of the hidden and observed sequence, whereby it is assumed that the probability distribution of each hidden symbol y_k depends solely on the value of the previous symbol in the sequence y_{k-1} (this is the Markov assumption which is quantified by $P(y_k|y_{k-1})$). Furthermore, it is assumed that the probability distribution of the observed symbol x_k depends solely on the value of y_k (quantified by the emission probability distribution $P(x_k|y_k)$). For simplicity, we ignore the probability distribution of the first element of the hidden chain in this exposition. The MAP estimator predicts the hidden sequence $\mathbf{y}$ that is most

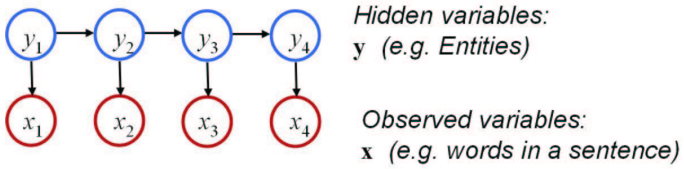


Figure 1: The graph of an HMM with $m = 4$.

likely given the observation sequence $\mathbf{x}$. In formulas:

$$\begin{aligned} h(\mathbf{x}) &= \arg\max_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \frac{P(\mathbf{y},\mathbf{x})}{P(\mathbf{x})} = \arg\max_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y})P(\mathbf{x}|\mathbf{y}), \\ &= \arg\max_{\mathbf{y}\in\mathcal{Y}} \prod_{k=2}^{m} P(y_k|y_{k-1}) \prod_{k=1}^{m} P(x_k|y_k), \\ &= \arg\max_{\mathbf{y}\in\mathcal{Y}} \left[\sum_{k=2}^{m} \log P(y_k|y_{k-1}) + \sum_{k=1}^{m} \log P(x_k|y_k) \right], \end{aligned}$$

where we made use of the fact that the argmax of a function is equal to the argmax of its logarithm.

Thus, to fully specify the HMM, one needs to consider all the transition probabilities (denoted t_{ij} for $i, j \in \Sigma_y$ for the transition from symbol i to j), and the emission probabilities (denoted e_{io} for the emission of symbol $o \in \Sigma_x$ by symbol $i \in \Sigma_y$). Using this notation, we can rewrite the prediction function as follows (with $I(\cdot)$ equal to one if the equalities between brackets hold):

$$\begin{aligned} h(\mathbf{x}) = \quad \arg\max_{\mathbf{y}\in\mathcal{Y}} \quad & \sum_{i,j\in\Sigma_y} \log(t_{ij}) \sum_{k=2}^{m} I(y_{k-1} = i, y_k = j) \\ & + \sum_{i\in\Sigma_y, o\in\Sigma_x} \log(e_{io}) \sum_{k=1}^{m} I(y_k = i, x_k = o). \end{aligned}$$

For simplicity of notation let us replace all logarithms of parameters t_{ij} and e_{io} by parameters θ_i summarized in a $d = n_h n_o + n_h^2$ dimensional parameter vector θ. Additionally, let us summarize the corresponding sufficient statistics $\sum_{k=2}^{m} I(y_{k-1} = i, y_k = j)$ and $\sum_{k=1}^{m} I(y_k = i, x_k = o)$ in a corresponding feature vector $\phi(\mathbf{x},\mathbf{y}) = [\phi_1(\mathbf{x},\mathbf{y})\ \phi_2(\mathbf{x},\mathbf{y})\ \ldots\ \phi_d(\mathbf{x},\mathbf{y})]^T$. (Note that these sufficient statistics count the number of occurrences of each specific transition and emission.) Then we can rewrite the prediction function in a linear form as required:

$$h_\theta(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \theta^T \phi(\mathbf{x},\mathbf{y}).$$

This prediction can be evaluated efficiently by means of the Viterbi algorithm. Note that in order to learn the parameters by means of maximum likelihood estimation, constraints are imposed to ensure that they represent log-probabilities. In order to arrive at a DLA that operates in the same setting, it suffices to ignore these constraints, and to minimize an appropriate empirical risk subject to some regularization, as outlined in Section 2.2. Moreover relaxing also the Markov assumption the proposed formulation can be extended to the case of arbitrary features. In general in fact the

vector $\phi(\mathbf{x},\mathbf{y})$ contains not only statistics associated to transition and emission probabilities but also any feature that reflects the properties of the objects represented by the nodes of the HMM. For example in most of the natural language processing tasks, feature vectors also contain information about spelling properties of words. Sometimes also the so-called 'overlapping features' (Lafferty et al., 2001) are employed, which indicate relations between observations and some previous and future labels. Most of DLAs dealing with this task have proceeded in this way (McCallum et al., 2000; Lafferty et al., 2001; Collins, 2002a; Altun et al., 2003a,b; Taskar et al., 2003).

2.3.2 SEQUENCE ALIGNMENT LEARNING

As second case studied, we consider the problem of learning how to align sequences: given as training examples a set of correct pairwise global alignments, find the parameter values that ensure sequences are optimally aligned. This task is also known as inverse parametric sequence alignment problem (IPSAP) and since its introduction in Gusfield et al. (1994), it has been widely studied (Gusfield and Stelling, 1996; Kececioglu and Kim, 2006; Joachims et al., 2005; Pachter and Sturmfels, 2004; Sun et al., 2004).

Consider two strings S_1 and S_2 of lengths n_1 and n_2 respectively. The strings are ordered sequences of symbols $s_i \in \mathcal{S}$, with $\mathcal{S}$ a finite alphabet of size $n_{\mathcal{S}}$. In case of biological applications, for DNA sequences the alphabet contains the symbols associated with nucleotides ($\mathcal{S} = \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}$), while for amino acids sequences the alphabet is $\mathcal{S} = \{\mathtt{A},\mathtt{R},\mathtt{N},\mathtt{D},\mathtt{C},\mathtt{Q},\mathtt{E},\mathtt{G},\mathtt{H},\mathtt{I},\mathtt{L},\mathtt{K},\mathtt{M},\mathtt{F},\mathtt{P},\mathtt{S},\mathtt{T},\mathtt{W},\mathtt{Y},\mathtt{V}\}$.

An alignment of two strings S_1 and S_2 of lengths n_1 and n_2 is defined as a pair of sequences T_1 and T_2 of equal length $n \geq n_1, n_2$ that are obtained by taking S_1 and S_2 respectively and inserting symbols $-$ at various locations in order to arrive at strings of length n. Two symbols in T_1 and T_2 are said to correspond if they occur at the same location in the respective string. If corresponding symbols are equal, this is called a *match*. If they are not equal, this is a *mismatch*. If one of the symbols is a $-$, this is called a *gap*.

With each possible match, mismatch or gap a score is attached. To quantify these scores, three score parameters can be used: one for matches (θ_m), one for mismatches (θ_s), and one for gaps (θ_g). In analogy with the notation in this paper, the pair of given sequences S_1 and S_2 represent the input variable $\mathbf{x}$ while their alignment is the output $\mathbf{y}$. The score of the global alignment is defined as the sum of this score over the length of T_1 and T_2, that is, as a linear function of the alignment parameters:

$$s_\theta(\mathbf{x},\mathbf{y}) = \theta^T \phi(\mathbf{x},\mathbf{y}) = \theta_m m + \theta_s s + \theta_g g$$

where $\phi(\mathbf{x},\mathbf{y}) = [m\, s\, g]^T$ and m, s and g represent the number of matches, mismatches and gaps in the alignment. Fig. 2 depicts a pairwise alignment between two sequences and the associated path in the alignment graph. The number N of all possible alignments between S_1 and S_2 is clearly exponential in the size of the two strings. However, an efficient DP algorithm for computing the alignment with maximal score $\bar{\mathbf{y}}$ is known in literature: the Needleman-Wunsch algorithm (Needleman, 1970).

The scoring models presented above consider a local form of gap penalty: the gap penalty is fixed independently of the other gaps in the alignment. However for biological reasons it is often preferable to consider an affine function for gap penalties, that is to assign different costs if the gap starts (gap opening penalty θ_o) in a given position or if it continues (gap extension penalty θ_e). Then the score of an alignment is:

$$s_\theta(\mathbf{x},\mathbf{y}) = \theta_m m + \theta_s s + \theta_o o + \theta_e e$$

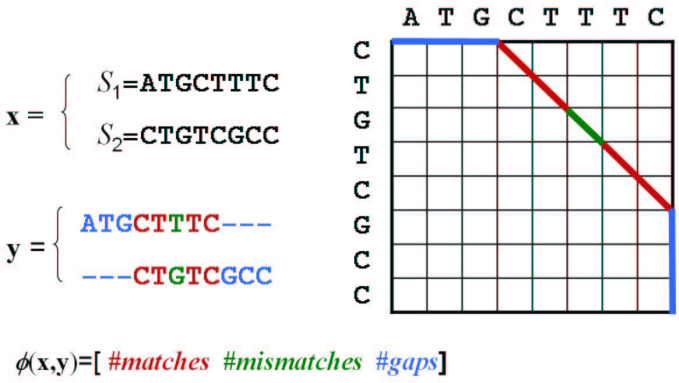


Figure 2: An alignment $\mathbf{y}$ between two sequences S_1 and S_2 can be represented by a path in the alignment graph.

where m, s, o and e represent the number of match, mismatch, gap openings and gap extensions respectively and θ_m, θ_s, θ_o, θ_e are the associated costs. As before we can define the vectors $\theta = [\theta_m\ \theta_s\ \theta_o\ \theta_e]^T$ and $\phi(\mathbf{x},\mathbf{y}) = [m\ s\ o\ e]^T$. Therefore the score is still a linear function of the parameters and the prediction can be computed by a DP algorithm.

More often a different model is considered where a (symmetric) scoring matrix specifies different score values for each possible pair of symbols. In general there are $d = \frac{n_S(n_S+1)}{2}$ different parameters in θ associated with the symbols of the alphabet plus two additional ones corresponding to the gap penalties. This means that to align sequences of amino acids we have 210 parameters to determine plus other 2 parameters for gap opening and gap extension. We denote with z_{jk} the number of pairs where a symbol of T_1 is j and it corresponds to a symbol k in T_2. Again the score is a linear function of the parameters:

$$s_\theta(\mathbf{x},\mathbf{y}) = \sum_{j \geq k} \theta_{jk} z_{jk} + \theta_o o + \theta_e e$$

and the optimal alignment is computed by the Needleman-Wunsch algorithm.

2.3.3 LEARNING TO PARSE

In learning to parse the input $\mathbf{x}$ is given by a sequence, and the output is given by its associated parse tree according to a context free grammar. Usually weighted context-free grammars (WCFGs) (Manning and Schetze, 1999) are used to approach this problem. Learning to parse has been already studied as a particular instantiation of structured output learning, both in natural language processing applications (Tsochantaridis et al., 2005; Taskar et al., 2004) and in computational biology for RNA secondary structure alignment (Sato and Sakakibara, 2005) and prediction (Do et al., 2006). In this paper we consider the latter and we use WCFGs to model the structure of RNA sequences. Two examples of RNA secondary structure for two sequences are shown in Fig. 3.

A WCFG is defined as five tuples $(\Upsilon, \Sigma_x, R, S, \theta)$, where $\Upsilon = \{\Upsilon_1, \ldots, \Upsilon_{|\Upsilon|}\}$ is a set of nonterminals, $\Sigma_x = \{X_1, \ldots, X_{|\Sigma_x|}\}$ is a set of terminals, $R = \{\Upsilon_i \rightarrow \alpha | \Upsilon_i \in \Upsilon, \alpha \in (\Upsilon \cup \Sigma_x)^*\}$ is a set of rules,

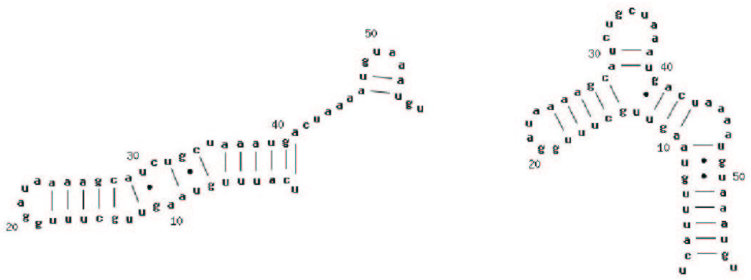


Figure 3: Two examples of RNA secondary structures for two sequences of the Rfam database (Griffiths-Jones et al., 2003).

$S \in \Upsilon$ is the starting symbol, and θ is a set of weights. We use rules of the forms $\Upsilon_i \to X$, $\Upsilon_i \to \Upsilon_j \Upsilon_k$, $\Upsilon_i \to X \Upsilon_j X'$, and $\Upsilon_i \to \Upsilon_{j'}$ ($j' > i$). R is also indexed by an ordering $\{r_1, \ldots, r_{|R|}\}$ and $d = |R|$. Each node in the parse tree $\mathbf{y}$ corresponds to a grammar rule and each weight $\theta_i \in \theta$ is associated with a rule $r_i \in R$. Given a sequence $\mathbf{x}$ and an associated parse tree $\mathbf{y}$ we can define a feature vector $\phi(\mathbf{x}, \mathbf{y})$ which contains a count of the number of occurrences of each of the rules in the parse tree $\mathbf{y}$. Given a parameter vector θ, the prediction function $h_\theta(\mathbf{x})$ is computed by finding the best parse tree. For SCFGs, this can be done efficiently with the Cocke-Younger-Kasami (CYK) algorithm (Younger, 1967).

3. Computing the Moments of the Scoring Function

An interesting corollary of the proposed structured output approach based on linear scoring functions is that certain statistics of the score $s(\mathbf{x}, \mathbf{y})$ can be expressed as function of the parameter vector θ. More specifically given an observed vector $\mathbf{x}$, we can consider the first order moment or mean $M_{1,\theta}(\mathbf{x})$ and the centered second order moment or covariance $M_{2,\theta}(\mathbf{x})$ of the scores along all possible N output variables $\mathbf{y}_j$. It is straightforward to see that $M_{1,\theta}(\mathbf{x})$ is a linear function of θ, that is,

$$\begin{aligned} M_{1,\theta}(\mathbf{x}) &\triangleq \frac{1}{N}\sum_{j=1}^{N} s_\theta(\mathbf{x}, \mathbf{y}_j) \\ &= \theta^T \frac{1}{N}\sum_{j=1}^{N} \phi(\mathbf{x}, \mathbf{y}_j) \\ &= \theta^T \mu \end{aligned}$$

with $\mu = [\mu_1 \ldots \mu_d]^T = \frac{1}{N}\sum_{j=1}^{N} \phi(\mathbf{x}, \mathbf{y}_j)$. Similarly, for the covariance:

$$\begin{aligned} M_{2,\theta}(\mathbf{x}) &\triangleq \frac{1}{N}\sum_{j=1}^{N} (s_\theta(\mathbf{x}, \mathbf{y}_j) - M_{1,\theta}(\mathbf{x}))^2 \\ &= \theta^T \left(\frac{1}{N}\sum_{j=1}^{N} (\phi(\mathbf{x}, \mathbf{y}_j) - \mu)(\phi(\mathbf{x}, \mathbf{y}_j) - \mu)^T \right) \theta \end{aligned}$$

$$= \quad \theta^T C \theta.$$

The matrix C is a matrix with elements:

$$\begin{aligned} c_{pq} &= \frac{1}{N}\sum_{j=1}^{N}(\phi_p(\mathbf{x},\mathbf{y}_j)-\mu_p)(\phi_q(\mathbf{x},\mathbf{y}_j)-\mu_q) \\ &= \frac{1}{N}\sum_{j=1}^{N}\left(\phi_p(\mathbf{x},\mathbf{y}_j)\phi_q(\mathbf{x},\mathbf{y}_j)\right)-\mu_p\mu_q = \nu_{pq}-\mu_p\mu_q \end{aligned} \tag{4}$$

where $1 \leq p,q \leq d$.

3.1 Magic Moments

It should be clear that in practical structured output learning problems the number N of possible output vectors associated to a given input $\mathbf{x}$ can be massive. At first sight, this leaves little hope that the above sums can ever be computed for realistic problems. However, it turns out that the same ideas that allow one to perform inference in PGMs allow one to compute these sums efficiently using DP, be it with somewhat more complicated recursions.

The underlying ideas to derive the recursions for μ are based on the commutativity of the semiring that is used in the Viterbi (or more generally the max-product and related algorithms) in PGMs. In particular, this recursion is used in various forms:

$$E\left\{\sum_{i=1}^{k} a_i\right\} \quad = \quad E\left\{\sum_{i=1}^{k-1} a_i\right\} + E\left\{a_k\right\},$$

where the expectations are jointly over independent random variables a_i. For the recursions for the second order moment (which can be used to compute the centered second order moment as shown in (4)), the following recursive expression is applied in different variations:

$$E\left\{\left(\sum_{i=1}^{k} a_i\right)^2\right\} \quad = \quad E\left\{\left(\sum_{i=1}^{k-1} a_i\right)^2\right\} + 2E\left\{a_k \sum_{i=1}^{k-1} a_i\right\} + E\left\{a_k^2\right\},$$

where again the expectations are jointly over independent random variables a_i. Note that the middle term on the right hand side is computed by previous iterations for the first order moment. For concreteness, we will now consider separately the three illustrative scenarios introduced above.

3.2 Sequence Labeling Learning

Given a fixed input sequence $\mathbf{x}$, we show here for the sequence labeling example that the elements of μ and C can be computed exactly and efficiently by dynamic programming routines.

We first consider the vector μ and construct it in a way that the first $n_h n_o$ elements contain the mean values associated with the emission probabilities and the remaining n_h^2 elements correspond to transition probabilities. Each value of μ can be determined by Algorithm 1.

In the emission part for each element a $n_h \times m$ dynamic programming table μ^e_{pq} is considered. The index p denotes the hidden state ($1 \leq p \leq n_h$) and q refers to the observation ($1 \leq q \leq n_o$). For example the first component of μ corresponds to the DP table μ^e_{11}. In practice each cell of the DP

table correspond to a node of the HMM trellis. At the same time another $n_h \times m$ DP table, denoted by π, is considered and filled in a way that each element $\pi(i,j)$ contains the number of all possible paths in the HMM trellis terminating at position (i,j). Then a recursive relation is considered to compute each element $\mu^e_{pq}(i,j)$, $\forall 1 \leq j \leq m$, $\forall 1 \leq i \leq n_h$. Basically at step (i,j) the mean value $\mu^e_{pq}(i,j)$ is given summing the occurrences of emission probabilities e_{pq} at the previous steps (e.g., $\sum_i \mu^e_{pq}(i,j-1)\pi(i,j-1)$) with the number of paths in the previous steps (if the current observation x_j is q and the current state y_j is p) and dividing this quantity by $\pi(i,j)$.

In a similar way the mean values associated to the transition probabilities are computed. Dynamic programming tables μ^t_{pz}, $1 \leq p,z \leq n_h$ are filled with recursive formulas in Algorithm 4 in appendix E.

Analogously the elements of the covariance matrix C can be obtained. We have five sets of values: variances of emission probabilities (c^e_{pq}, $1 \leq p \leq n_h, 1 \leq q \leq n_o$), variances of transition probabilities (c^t_{pz}, $1 \leq p,z \leq n_h$), covariances of emission probabilities ($c^e_{pqp'q'}$, $1 \leq p,p' \leq n_h, 1 \leq q,q' \leq n_o$), covariances of transition probabilities ($c^t_{pzp'z'}$, $1 \leq p,p',z,z' \leq n_h$) and mixed covariances ($c^{et}_{pqp'z}$, $1 \leq p,p',z \leq n_h, 1 \leq q \leq n_o$). To determine each of them we consider (4) and we compute the values v^e_{pq}, v^t_{pz}, $v^e_{pqp'q'}$, $v^t_{pzp'z'}$ and $v^{et}_{pqp'z}$ since the mean values are already known. This computation is again performed following Algorithm 1 but with recursive relations given in Algorithm 4, in appendix E (the number 5, 11, 12 in Algorithm 4 are meant to indicate the lines of Algorithm 1 where the formulas must be inserted).

Algorithm 1 Computation of μ^e_{pq} for sequence labeling learning

Input: $\mathbf{x} = (x_1, x_2, ..., x_m)$, p, q.

for $i = 1$ **to** n_h
    $\pi(i,1) := 1$
    if $q = x_1 \wedge p = i$, **then** $\mu^e_{pq}(i,1) := 1$
end
for $j = 2$ **to** m
    for $i = 1$ **to** n_h
        $M := 0$
        $\pi(i,j) := \sum_i \pi(i,j-1)$
        if $q = x_j \wedge p = i$, **then** $M := 1$
        $\mu^e_{pq}(i,j) := \frac{\sum_i (\mu^e_{pq}(i,j-1)+M)\pi(i,j-1)}{\pi(i,j)}$
    end
end

Output: $\frac{\sum_i \mu^e_{pq}(i,m)\pi(i,m)}{\sum_i \pi(i,m)}$

3.2.1 COMPUTATIONAL COST ANALYSIS

At first sight the calculation of μ and C requires running a DP algorithm like Algorithm 1 d times for μ and d^2 times for C. Hence the overall computational cost seems to depend strongly on d. However, most of the DP routines are redundant since many cells of μ and C have the same values. In fact, the following can be shown:

Proposition 3 *The number of dynamic programming routines required to calculate* μ and C *increases linearly with the size of the observation alphabet.*

An outline of proof can be found in appendix A.

Algorithms 1 and 4 assume that the HMM is 'fully connected', that is, transitions are allowed from and to any possible hidden states and every symbol can be emitted in every state. However, this condition is often not satisfied in practical applications. We should point out that their adaptation for such situations is straightforward and involves computing only sums that correspond to allowed paths in the DP table. In this case the number of distinct parameters as well as the computational cost increases with respect to complete models. However this effect may be offset by the fact that each DP becomes less time consuming. Furthermore the mean and the covariance values associated to transition probabilities are independent from observations. To calculate them a closed form expression can be used without the need of running any DP routine.

Moreover usually in most applications the size of the observation alphabet (for example the size of the dictionary in a natural language processing system) is very large while the sequences to be labeled are short. This means that the number of distinct observations in each sequence $\mathbf{x}$ is much lower than n_o. In such cases the number of different values in μ and C scales linearly with it.

We point out that the proposed algorithm can be easily extended to the case of arbitrary features in the vector $\phi(\mathbf{x},\mathbf{y})$ (not only those associated with transition and emission probabilities). To compute μ and C in these situations the derivation of appropriate formulas similar to those of μ^e_{pq}, c^e_{pq} and $c^{et}_{pqp'z}$ is straightforward.

3.2.2 ESTIMATING μ AND C BY RANDOM SAMPLING

Still, the computational cost increases with the number of features since for HMMs that are not 'fully connected', it may occur that the number of different values in the matrix C scales quadratically with the observations alphabet size n_o. However we show that in this case accurate and efficient approximation algorithms can be used to obtain close estimates of the mean and the variance values with a significantly reduced computational cost. This can be achieved by considering a finite subsample of all possible values for the output $\mathbf{y}$, rather than using the DP approaches. This comment holds generally for all learning problems considered in this paper, and we come back to this in the theoretical discussion in 6.1 as well as in the experimental results in 5 to support this claim empirically.

3.3 Sequence Alignment Learning

For the sequence alignment learning task we consider separately the three parameter model, the model with affine gap penalties and the model with substitution matrices.

3.3.1 THE SIMPLEST SCORING SCHEME: MATCH, MISMATCH, GAP

In this model the vector $\mu = [\mu_m\ \mu_s\ \mu_g]^T$ contains the average number of matches, mismatches and gaps computed considering all possible alignments. Its elements can be obtained using Algorithm 2. In a nutshell, the algorithm works as follows. First, a matrix π is filled. Every cell $\pi(i,j)$ contains the number of all possible alignments between two prefixes of the strings S_1 and S_2. In fact each alignment corresponds to a path in the alignment graph associated with the DP matrix. At the same time the DP tables for μ_m, μ_s and μ_g are gradually filled according to appropriate recursive relations.

For example each element $\mu_m(i,j)$ is computed dividing the total number of matches by the number of alignments $\pi(i,j)$. If a match occur in position (i,j) ($M=1$) the total number of matches at step (i,j) is obtained adding to the number of matches in the previous steps ($\mu_m(i,j-1)\pi(i,j-1)$, $\mu_m(i-1,j-1)\pi(i-1,j-1)$ and $\mu_m(i-1,j)\pi(i-1,j)$) $\pi(i-1,j-1)$ times a match. Once the algorithm is terminated, the mean values can be read in the cells $\mu_m(n_1,n_2)$, $\mu_s(n_1,n_2)$ and $\mu_g(n_1,n_2)$.

The covariance matrix C is the 3×3 matrix with elements c_{pq}, $p,q\in\{m,s,g\}$ and it is symmetric ($c_{sg}=c_{gs}$, $c_{mg}=c_{gm}$, $c_{sm}=c_{ms}$). Each value c_{pq} can be obtained considering (4) and computing the associated values v_{pq} with appropriate recursive relations (see Algorithm 2).

3.3.2 AFFINE GAP PENALTIES

As before we can define the vector $\mu=[\mu_m\ \mu_s\ \mu_o\ \mu_e]^T$ and the covariance matrix C as the 4×4 symmetric matrix with elements c_{pq} with $p,q\in\{m,s,o,e\}$. The values of μ and C are computed with DP. In particular μ_m, μ_s, v_{mm}, v_{ms} and v_{ss} are calculated as above, while the other values are obtained with the formulas in Algorithm 5 in appendix E. The terms v_{se} and v_{so} are missing since they can be calculated with the same formulas of v_{me} and v_{mo} simply changing M with $1-M$ and μ_m with μ_s. Note that in some situations for low values of (i,j) some terms are not defined (i.e., $\pi(i,j-3)$ when $j=2$). In such situations they must be ignored in the computation.

3.3.3 EXTENSION TO A GENERAL SCORING MATRIX

The formulas illustrated in the previous paragraphs can be extended to the case of a general substitution matrix with minor modifications. Concerning the mean values, μ_o and μ_e are calculated as before. For the others it is:

$$\mu_{z_{pq}}(i,j):=\frac{\mu_{z_{pq}}(i-1,j)\pi(i-1,j)+\mu_{z_{pq}}(i,j-1)\pi(i,j-1)+(\mu_{z_{pq}}(i-1,j-1)+M)\pi(i-1,j-1)}{\pi(i,j)}$$

where $M=1$ when two corresponding symbols in the alignment are equal to p and q or vice versa with $p,q\in\mathcal{S}$. The matrix C is a symmetric matrix 212×212. The values v_{eo}, v_{ee} and v_{oo} are calculated as above. The derivation of formulas for $v_{z_{pq}z_{p'q'}}$ is straightforward from v_{ms} considering the appropriate values for M and the mean values. The formulas for v_{zo} and v_{ze} follow with minor modification from v_{mo} and v_{me}.

3.4 Learning to Parse

For a given input string $\mathbf{x}$, let μ_p and c_{pq} be the mean of occurrences of rule p and the covariance between the numbers of occurrences of rules p and q, respectively, that is, the elements of μ and C. The following relations hold:

$$\begin{aligned}\mu_p &= \frac{1}{N}\sum_{j=1}^{N}\phi_p(\mathbf{x},\mathbf{y}_j)=\frac{1}{N}\psi_p,\\ nc_{pq} &= \frac{1}{N}\sum_{j=1}^{N}\left(\phi_p(\mathbf{x},\mathbf{y}_j)\phi_q(\mathbf{x},\mathbf{y}_j)\right)-\mu_p\mu_q=\frac{1}{N}\gamma_{pq}-\mu_p\mu_q,\end{aligned}$$

where N is the number of all possible parse trees associated to $\mathbf{x}$, ψ_p is the number of occurrences of the rule p in all the parse tree $\mathbf{y}_j$ given $\mathbf{x}$, and γ_{pq} denotes the cooccurrences of p and q.

To compute C and μ an algorithm based on a bottom-up dynamic programming can be developed. Similarly to sequence labeling three types of recurrence equations must be defined: one to

Algorithm 2 Computation of μ and C with matches, mismatches and gaps.

Input: a pair of sequences S_1 and S_2.

$\pi(0,0) := 1$
$\mu_m(0,0) = \mu_s(0,0) = \mu_g(0,0) := 0$
$v_{mm}(0,0) = v_{ms}(0,0) = v_{ss}(0,0) = v_{sg}(0,0) = v_{mg}(0,0) = v_{gg}(0,0) := 0$
for $i = 1 : n_1$
    $\pi(i,0) := 1$
    $\mu_g(i,0) := \mu_g(i-1,0) + 1$
    $v_{gg}(i,0) := v_{gg}(i-1,0) + 2\mu_g(i-1,0) + 1$
end
for $j = 1 : n_2$
    $\pi(0,j) := 1$
    $\mu_g(0,j) := \mu_g(0,j-1) + 1$
    $v_{gg}(0,j) := v_{gg}(0,j-1) + 2\mu_g(0,j-1) + 1$
end
for $i = 1 : n_1$
    for $j = 1 : n_2$
        $\pi(i,j) := \pi(i-1,j-1) + \pi(i,j-1) + \pi(i-1,j)$
        if $s_1(i) = s_2(j)$ **then** $M := 1$ **else** $M := 0$
        $\mu_m(i,j) := \frac{\mu_m(i-1,j)\pi(i-1,j) + \mu_m(i,j-1)\pi(i,j-1) + (\mu_m(i-1,j-1)+M)\pi(i-1,j-1)}{\pi(i,j)}$
        $\mu_s(i,j) := \frac{\mu_s(i-1,j)\pi(i-1,j) + \mu_s(i,j-1)\pi(i,j-1) + (\mu_s(i-1,j-1)+(1-M))\pi(i-1,j-1)}{\pi(i,j)}$
        $\mu_g(i,j) := \frac{\mu_g(i-1,j)+1)\pi(i-1,j) + (\mu_g(i,j-1)+1)\pi(i,j-1) + \mu_g(i-1,j-1)\pi(i-1,j-1)}{\pi(i,j)}$
        $v_{mm}(i,j) := \frac{1}{\pi(i,j)}(v_{mm}(i-1,j)\pi(i-1,j) + v_{mm}(i,j-1)\pi(i,j-1)$
            $+(v_{mm}(i-1,j-1) + 2M\mu_m(i-1,j-1) + M)\pi(i-1,j-1))$
        $v_{ss}(i,j) := \frac{1}{\pi(i,j)}(v_{ss}(i-1,j)\pi(i-1,j) + v_{ss}(i,j-1)\pi(i,j-1)$
            $+(v_{ss}(i-1,j-1) + 2(1-M)\mu_s(i-1,j-1) + (1-M))\pi(i-1,j-1))$
        $v_{gg}(i,j) := \frac{1}{\pi(i,j)}(v_{gg}(i-1,j) + 2\mu_g(i-1,j) + 1)\pi(i-1,j)$
            $+(v_{gg}(i,j-1) + 2\mu_g(i,j-1) + 1)\pi(i,j-1) + v_{gg}(i-1,j-1)\pi(i-1,j-1))$
        $v_{mg}(i,j) := \frac{1}{\pi(i,j)}(v_{mg}(i-1,j) + \mu_m(i-1,j))\pi(i-1,j) + (v_{mg}(i,j-1)$
            $+\mu_m(i,j-1))\pi(i,j-1) + (v_{mg}(i-1,j-1) + M\mu_g(i-1,j-1))\pi(i-1,j-1))$
        $v_{sg}(i,j) := \frac{1}{\pi(i,j)}(v_{sg}(i-1,j) + \mu_s(i-1,j))\pi(i-1,j) + (v_{sg}(i-1,j-1)$
            $+(1-M)\mu_g(i-1,j-1) + (v_{sg}(i,j-1) + \mu_s(i,j-1))\pi(i,j-1))\pi(i-1,j-1))$
        $v_{ms}(i,j) := \frac{1}{\pi(i,j)}(v_{ms}(i-1,j)\pi(i-1,j) + v_{ms}(i,j-1)\pi(i,j-1)$
            $+(v_{ms}(i-1,j-1) + M\mu_s(i-1,j-1) + (1-M)\mu_m(i-1,j-1))\pi(i-1,j-1))$
    end
end

Output: $\mu_m(n_1,n_2)$, $\mu_s(n_1,n_2)$, $\mu_g(n_1,n_2)$,
    $c_{mm}(n_1,n_2) := v_{mm}(n_1,n_2) - \mu_m(n_1,n_2)^2$,
    $c_{ss}(n_1,n_2) := v_{ss}(n_1,n_2) - \mu_m(n_1,n_2)^2$,
    $c_{gg}(n_1,n_2) := v_{gg}(n_1,n_2) - \mu_m(n_1,n_2)^2$,
    $c_{ms}(n_1,n_2) := v_{ms}(n_1,n_2) - \mu_m(n_1,n_2)\mu_s(n_1,n_2)$,
    $c_{mg}(n_1,n_2) := v_{mg}(n_1,n_2) - \mu_m(n_1,n_2)\mu_g(n_1,n_2)$,
    $c_{sg}(n_1,n_2) := v_{sg}(n_1,n_2) - \mu_s(n_1,n_2)\mu_g(n_1,n_2)$

compute the number of parse trees N, another the number of occurrences ψ of each parameter, and the latter the number of cooccurrences γ of each pair of parameters.

For a given input string $\mathbf{x} = (x_1\ x_2 \ldots\ x_m)$, x_s denotes the s-th symbol of $\mathbf{x}$, and $x_{s|t}$ the substring from the s-th symbol to the t-th symbol. We count the number of possible trees N given $\mathbf{x}$ with a DP algorithm such as the CYK algorithm. We use two types of auxiliary variables, $\pi(s,t,\Upsilon_i)$ and $\pi(s,t,\Upsilon_i,\alpha)$ which are the number of possible parse trees whose root is Υ_i for substring $x_{s|t}$, and the number of possible parse trees whose root is applied to rule $\Upsilon_i \to \alpha$ for substring $x_{s|t}$, where $(\Upsilon_i \to \alpha) \in R$.

Then $\pi(s,t,\Upsilon_i)$ is calculated as follows:

$$\pi(s,t,\Upsilon_i) = \sum_{\alpha:(\Upsilon_i\to\alpha)\in\Upsilon} \pi(s,t,\Upsilon_i,\alpha),$$

where:

$$\pi(s,t,\Upsilon_i,\alpha) \quad = \quad \begin{cases} 1 & \alpha = X \in \Sigma_x,\ s=t, \text{ and } X = x_s, \\ \sum_{r=s}^{t-1} \pi(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \alpha = \Upsilon_{k_1}\Upsilon_{k_2} \text{ and } s<t, \\ \pi(s,t,\Upsilon_k) & \alpha = \Upsilon_k, \\ \pi(s+1,t-1,\Upsilon_k) & \alpha = X\Upsilon_k X',\ X = x_s, \text{ and } X' = x_t, \\ 0 & \text{otherwise.} \end{cases}$$

Upon completion of the recursion, $N = \pi(1,m,S)$ is the number of all possible parse trees given $\mathbf{x}$.

We then count the number of occurrences of each rule in all possible parse trees. $\psi_p(s,t,\Upsilon_i)$ denotes the number of occurrences of rule p in all possible parse trees whose root is Υ_i for $x_{s|t}$. We compute $\psi_p(s,t,\Upsilon_i)$ as follows:

$$\psi_p(s,t,\Upsilon_i) = \sum_{\alpha:(\Upsilon_i\to\alpha)\in\Upsilon} \psi_p(s,t,\Upsilon_i,\alpha),$$

where:

$$\begin{aligned} &\psi_p(s,t,\Upsilon_i,\alpha) \\ &= \begin{cases} 1 & \alpha = X = x_s,\ s=t, \\ & \text{and } p = \Upsilon_i \to X, \\ \sum_{r=s}^{t-1} (\psi_p(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \\ \quad +\pi(s,r,\Upsilon_{k_1})\psi_p(r+1,t,\Upsilon_{k_1})) & \\ \quad +I(p,\Upsilon_i\to\alpha)\pi(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \alpha = \Upsilon_{k_1}\Upsilon_{k_2}, \text{ and } s<t, \\ \psi_p(s,t,\Upsilon_k) + I(p,\Upsilon_i\to\alpha)\pi(s,t,\Upsilon_k) & \alpha = \Upsilon_k, \\ \psi_p(s+1,t-1,\Upsilon_k) + I(p,\Upsilon_i\to\alpha)\pi(s+1,t-1,\Upsilon_k) & \alpha = X\Upsilon_k X' \text{ and } s+1<t, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

with $I(p,\Upsilon_i\to\alpha) = 1$ if $p = (\Upsilon_i \to \alpha)$, otherwise it is $I(p,\Upsilon_i\to\alpha) = 0$. Then, $\psi_p(1,m,S)$ is the number of occurrences of p in all parse trees given $\mathbf{x}$.

We count the number of cooccurrences $\gamma_{pq}(s,t,\Upsilon_i)$ in each pair p and q of rules. $\gamma_{pq}(s,t,\Upsilon_i,\alpha)$ denotes the number of cooccurrences in all possible parse trees whose root is Υ_i for $x_{s|t}$. We calculate $\gamma_{pq}(s,t,\Upsilon_i)$ as follows:

$$\gamma_{pq}(s,t,\Upsilon_i) = \sum_{\alpha:(\Upsilon_i\to\alpha)\in\Upsilon} \gamma_{pq}(s,t,\Upsilon_i,\alpha),$$

where:

$$
\gamma_{pq}(s,t,\Upsilon_i,\alpha)
$$

$$
= \begin{cases}
0 & \alpha = X \text{ and } p \neq q, \\
1 & \alpha = X \text{ and } p = q = (\Upsilon_i \rightarrow \alpha), \\
\sum_{r=s}^{t-1} \Big(\gamma_{pq}(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) & \\
+\pi(s,r,\Upsilon_{k_1})\gamma_{pq}(r+1,t,\Upsilon_{k_2}) & \\
+\psi_p(s,r,\Upsilon_{k_1})\psi_q(r+1,t,\Upsilon_{k_2}) & \\
+\psi_q(s,r,\Upsilon_{k_1})\psi_p(r+1,t,\Upsilon_{k_2}) & \\
+I(p,\Upsilon_i \rightarrow \alpha) f(p,s,r,t,\Upsilon_{k_1},\Upsilon_{k_2}) & \\
+I(q,\Upsilon_i \rightarrow \alpha) f(q,s,r,t,\Upsilon_{k_1},\Upsilon_{k_2}) & \\
+I(p,\Upsilon_i \rightarrow \alpha) I(q,\Upsilon_i \rightarrow \alpha)\pi(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) \Big) & \alpha = \Upsilon_{k_1}\Upsilon_{k_2} \text{ and } s < t, \\
\gamma_{pq}(s,t,\Upsilon_k) & \\
+I(p,\Upsilon_i \rightarrow \alpha)\psi_q(s,t,\Upsilon_i) + I(q,\Upsilon_i \rightarrow \alpha)\psi_p(s,t,\Upsilon_i) & \\
+I(p,\Upsilon_i \rightarrow \alpha) I(q,\Upsilon_i \rightarrow \alpha)\pi(s,t,\Upsilon_i) & \alpha = \Upsilon_k, \\
\gamma_{pq}(s+1,t-1,\Upsilon_k) & \\
+I(p,\Upsilon_i \rightarrow \alpha)\psi_q(s+1,t-1,\Upsilon_k) & \\
+I(q,\Upsilon_i \rightarrow \alpha)\psi_p(s+1,t-1,\Upsilon_k) & \\
+I(p,\Upsilon_i \rightarrow \alpha) I(q,\Upsilon_i \rightarrow \alpha)\pi(s+1,t-1,\Upsilon_k) & \alpha = X\Upsilon_k X',\ s+1 < t, \\
 & x_s = X, \text{ and } x_t = X', \\
0 & \text{otherwise}
\end{cases}
$$

with $f(p,s,r,t,\Upsilon_{k_1},\Upsilon_{k_2}) = \psi_p(s,r,\Upsilon_{k_1})\pi(r+1,t,\Upsilon_{k_2}) + \pi(s,r,\Upsilon_{k_1})\psi_p(r+1,t,\Upsilon_{k_2})$. Finally, $\gamma_{pq}(1,m,S)$ is the number of cooccurrences of rules p and q in all parse trees given $\mathbf{x}$.

In the following section we discuss how we can use the computed first and second order statistics to define a suitable objective function which can be optimized for structured output learning tasks.

4. Moment-based Approaches to Structured Output Prediction

Suppose we have a training set of input-output pairs $\mathcal{T} = \{(\mathbf{x}^1,\overline{\mathbf{y}}^1),(\mathbf{x}^2,\overline{\mathbf{y}}^2),\ldots,(\mathbf{x}^\ell,\overline{\mathbf{y}}^\ell)\}$. The task we consider is to find the parameter values θ such that the optimal given output variables $\overline{\mathbf{y}}^i$ can be reconstructed from $\mathbf{x}^i$, $\forall\ 1 \leq i \leq \ell$. We want to fulfill this task by defining a suitable objective function which is a convex function of the first and second order statistics we presented before. Based on this idea we introduce two possible approaches.

4.1 Training Sets of Size One

To give an intuition of the main idea behind both methods, we first analyze the situation where the training set in made of only one pair $(\mathbf{x},\overline{\mathbf{y}})$. In this situation, both methods are identical to each other.

The idea is to consider the distribution of the scores for all possible $\mathbf{y}$. We then define a measure of separation between the score of the correct training output, and the entire distribution of all scores for all possible outputs. More specifically, the objective function we propose is the difference between the score of the true output and the mean score of the distribution, divided by the square root of the variance as a normalization. Mathematically:

$$
\max_{\theta} \frac{s_\theta(\mathbf{x},\overline{\mathbf{y}}) - M_{1,\theta}(\mathbf{x})}{\sqrt{M_{2,\theta}(\mathbf{x})}} \quad = \quad \max_{\theta} \frac{\theta^T b}{\sqrt{\theta^T C \theta}} \tag{5}
$$

where $b = \phi(\mathbf{x}, \overline{\mathbf{y}}) - \mu$ is the difference between the feature vector associated to the optimal output and the average feature vector μ. Maximizing this objective over θ means that we search for a parameter vector θ that makes the score of the correct output $\overline{\mathbf{y}}$ as different as possible from the mean score, measured in number of standard deviations. This corresponds to a well known quantity in statistics: the Z-score. Given the distribution of all possible scores (i.e., given its mean and its variance), the Z-score of the correct pair $(\mathbf{x}, \overline{\mathbf{y}})$ is defined as the number of standard deviations its score $s(\mathbf{x}, \overline{\mathbf{y}})$ is away from the mean of the distribution.

The Z-score is an interesting measure of separation between the correct output and the bulk of all possible outputs corresponding to a given input. Under normality assumptions, it is directly equivalent to a *p*-value. Hence, maximizing the Z-score can be interpreted as maximizing the significance of the score of the correct pair: the larger the Z-score, the more significant it is, and the fewer other outputs would achieve a larger score. If the normality assumption is too unrealistic, one could still apply a (looser) Chebyshev tail bound to show that the number of scores that exceed the score of a large training output score $s_\theta(\mathbf{x}, \overline{\mathbf{y}})$ is small.

To quantify this connection between the Z-score of a training pair and the rank of its score among all other scores, we would like to introduce an alternative formulation for optimization problem (5).

Proposition 4 *Optimization problem (5) is equivalent to:*

$$\begin{array}{rl} \min_\theta & \frac{1}{N}\sum_{j=1}^N \xi_j^2 \\ s.t. & \theta^T\left(\phi(\boldsymbol{x},\overline{\boldsymbol{y}}) - \phi(\boldsymbol{x},\boldsymbol{y}_j)\right) = 1 + \xi_j \quad \forall j \end{array} \tag{6}$$

in the sense that it is optimized by the same value of θ *or a scalar multiple of it.*

Proof Substituting ξ_j from the constraint in the objective, the objective of optimization problem (6) is equivalent to:

$$\begin{aligned} & \frac{1}{N}\theta^T \sum_{j=1}^N (\phi(\mathbf{x},\overline{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j))(\phi(\mathbf{x},\overline{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j))^T\theta - \frac{2}{N}\theta^T \sum_{j=1}^N (\phi(\mathbf{x},\overline{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j)) + 1 \\ = \;& \frac{1}{N}\theta^T \sum_{j=1}^N (\mu - \phi(\mathbf{x},\mathbf{y}_j))(\mu - \phi(\mathbf{x},\mathbf{y}_j))^T\theta + \theta^T(\phi(\mathbf{x},\overline{\mathbf{y}}) - \mu)(\phi(\mathbf{x},\overline{\mathbf{y}}) - \mu)^T\theta \\ & -2(\phi(\mathbf{x},\overline{\mathbf{y}}) - \mu) + 1 \\ = \;& \theta^T C\theta + (\theta^T b - 1)^2. \end{aligned}$$

Hence, the optimization problem (6) is equivalent to:

$$\min_\theta \quad \theta^T C\theta + (\theta^T b - 1)^2.$$

Now, note that the objective in optimization problem (5) is invariant with respect to scaling of θ. Hence, we can fix the scale arbitrarily, and require $\theta^T b = 1$. The optimization problem then reduces to (using the monotonicity of the square root):

$$\begin{array}{rl} \min_\theta & \theta^T C\theta \\ \text{s.t.} & \theta^T b = 1. \end{array}$$

The optimality conditions of the former are $C\theta + bb^T\theta = b \Leftrightarrow C\theta = (1 - b^T\theta)b$, and the Lagrange optimality conditions of the latter are $C\theta = \lambda b$ with λ a Lagrange multiplier. Hence, both optimality conditions and optimization problems are equivalent in the sense that they are optimized by the

same θ up to a scaling factor. ■

The following interesting theorem now establishes the link between the relative ranking loss $\mathcal{L}_\theta^{RR}$ as defined in Section 2.2 and the above optimization problem.

Theorem 5 (Relative ranking loss upper bound) *Let us denote by $\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})$ the value of the objective of optimization problem (6) evaluated on training pair $(\boldsymbol{x},\bar{\boldsymbol{y}})$:*

$$\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}}) \quad = \quad \frac{1}{N}\sum_{j=1}^{N}\xi_i^2 = \frac{1}{N}\sum_{j=1}^{N}\left(\theta^T\left(\phi(\boldsymbol{x},\bar{\boldsymbol{y}})-\phi(\boldsymbol{x},\boldsymbol{y}_j)\right)-1\right)^2.$$

Then,

$$\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}}) \geq \mathcal{L}_\theta^{RR}(\boldsymbol{x},\bar{\boldsymbol{y}}).$$

(The RRU in the superscript stands for Relative Ranking Upper bound.)

Proof The rank of $s_\theta(\mathbf{x},\bar{\mathbf{y}})$ among all $s_\theta(\mathbf{x},\mathbf{y}_j)$ for all possible $\mathbf{y}_j$ is given by the number of $\mathbf{y}_j$ for which $\theta^T\left(\phi(\mathbf{x},\bar{\mathbf{y}})-\phi(\mathbf{x},\mathbf{y}_j)\right)\leq 0$. Hence, this is the number of times that $\xi_i \leq -1$ in optimization problem (6), such that the objective is at least as large as the rank divided by N, that is, the relative rank. ■

Additionally, we would like to point out that optimization problem (5) and equivalently (6) is also strongly connected to Fisher's discriminant analysis (FDA). Intuitively, maximizing our objective function corresponds to maximizing the distance between the mean of the distribution of the scores for all possible incorrect pairs and the 'mean' of the 'distribution' of the score for the single correct output, normalized by the sum of the standard deviations (note that one class reduces to one data point so the associated standard deviation is zero). Then (5) is equivalent to performing FDA when one class reduces to a single data point as defined by the correct training label.

4.2 Training Sets of General Sizes

Having introduced the main idea on the special case of a training set of size 1, we now turn back to the general situation where we are interested in computing the optimal parameter vector given a training set $\mathcal{T}$ of ℓ pairs of sequences. We will consider two different generalizations to which we refer as the Z-score based approach, and as structured output discriminant analysis (SODA).

4.3 Z-score Based Algorithm

In the first generalization, we will emphasize the Z-score interpretation. For training sets containing more than one input-output pair, we need to redefine the Z-score for a set of ℓ pairs of sequences $\mathcal{T}=\{(\mathbf{x}^1,\bar{\mathbf{y}}^1),(\mathbf{x}^2,\bar{\mathbf{y}}^2),\ldots,(\mathbf{x}^\ell,\bar{\mathbf{y}}^\ell)\}$. A natural way is to do this based on the global score: the sum of the scores for all sequence pairs in the set. Its mean is the sum of the means for all sequence pairs $(\mathbf{x}^i,\bar{\mathbf{y}}^i)$ separately, and can be summarized by $\bar{b}=\sum_i b_i$. Similarly, for the covariance matrix: $\bar{C}=\sum_i C_i$. Hence, the Z-score definition can naturally be extended to more than one input-output pair by using $\bar{b}$ and $\bar{C}$ instead of b and C in (5). In summary, extending the optimization problem (5) to the general situation of a given training set $\mathcal{T}$, the optimization problem we are interested in is:

$$\max_\theta \quad \frac{\theta^T\bar{b}}{\sqrt{\theta^T\bar{C}\theta}}. \tag{7}$$

The solution of (7) can be computed by simply solving the linear system $\bar{C}\theta = \bar{b}$, where $\bar{C}$ is a symmetric positive definite matrix. If $\bar{C}$ is not symmetric positive definite, regularization can be introduced in a straightforward way (similar as in FDA) by solving $(\bar{C} + \lambda I)\theta = \bar{b}$ instead. This effectively amounts to restricting the norm of θ to small values. Then the optimal parameter vector can be obtained extremely efficiently by using iterative methods such as the conjugate gradient method.

4.3.1 INCORPORATING THE HAMMING DISTANCE

A nice property of this approach is that it can be extended to take into account the Hamming distance between the output vectors. For each pair $(\mathbf{x}, \mathbf{y})$ we consider the score:

$$s(\mathbf{x}, \mathbf{y}) = \theta^T \phi(\mathbf{x}, \mathbf{y}) + \delta_H(\mathbf{y}, \bar{\mathbf{y}}) = \theta'^T \phi'(\mathbf{x}, \mathbf{y})$$

where we have defined the vectors $\theta'^T = \begin{bmatrix} \theta^T & 1 \end{bmatrix}$ and $\phi'(\mathbf{x}, \mathbf{y})^T = \begin{bmatrix} \phi(\mathbf{x}, \mathbf{y})^T & \delta_H(\mathbf{y}, \bar{\mathbf{y}}) \end{bmatrix}$. It is easy to verify that the associated optimization problem has the same form of (7) when the vectors θ' and ϕ' are considered. In practice the covariance matrix C is augmented with one column (and one row, since it is symmetric) containing the covariance values between the loss term and all the other parameters. We refer to this column as c_δ. Analogously the mean vector is augmented by one value (μ_δ) that represents the mean value of the terms $\delta_H(\mathbf{y}, \bar{\mathbf{y}})$ computed along all negative pseudoexamples. When the Hamming distance is adopted the computation of μ_δ and c_δ can be realized with DP algorithms. For example for sequence labeling learning Algorithm 1 is used with recursive relations similar to those in Algorithm 4.

4.3.2 Z-SCORE APPROACH WITH CONSTRAINTS

As a side remark, let us draw a connection with existing MM approaches such as described in Taskar et al. (2003) and in Tsochantaridis et al. (2005).

Their approach to structured output learning is to explicitly search for the parameter values θ such that the optimal hidden variables $\bar{\mathbf{y}}^i$ can be reconstructed from $\mathbf{x}^i$, $\forall 1 \leq i \leq \ell$. In formulas these conditions can be expressed as:

$$\theta^T \phi(\mathbf{x}^i, \bar{\mathbf{y}}^i) \geq \theta^T \phi(\mathbf{x}^i, \mathbf{y}^i_j) \quad \forall 1 \leq i \leq \ell \quad \forall 1 \leq j \leq N_i. \tag{8}$$

This set of constraints defines a convex set in the parameter space and its number is massive, due to the huge size of the output space. To obtain an optimal set of parameters θ that successfully fulfill (8) usually an optimization problem is formulated with these constraints, together with a suitable objective function. In MM approaches for example this objective function is typically chosen to be the squared norm of the parameter vector.

Interestingly, using the Z-score as objective function, we observe that most (and often all) of the constraints (8) are satisfied automatically, which often leads to a satisfactory result with a good generalization performance without considering the constraints explicitly.

However, in the cases where the result of (7) still violates some of the constraints and one wishes to avoid this, one can choose to impose these explicitly. The resulting optimization problem is still convex and it reduces to:

$$\begin{aligned} \min_\theta \quad & \theta^T \bar{C} \theta \\ \text{s.t.,} \quad & \theta^T \bar{b} \geq 1 \\ & \theta^T \phi(\mathbf{x}^i, \bar{\mathbf{y}}^i) \geq \theta^T \phi(\mathbf{x}^i, \mathbf{y}^i_j) \quad \forall 1 \leq i \leq \ell \quad \forall 1 \leq j \leq N_i. \end{aligned} \tag{9}$$

We have developed an incremental algorithm that implements problem (9), shown in Algorithm 3 (see Tsochantaridis et al., 2005, for a similar approach and a more detailed study). First a feasible solution is determined without adding any constraints. Then the following steps are repeated until convergence. For each training example, the most likely hidden variables are determined by a Viterbi-like algorithm. If its score is higher than the given one, the associated constraint is added to the set of constraints of the problem (9) and (9) is solved. The convergence is guaranteed from the convexity of the problem. Each added constraint provides the effect of restricting the feasible region.

Algorithm 3 Iterative algorithm to incorporate the active constraints.

Input: The training set $\mathcal{T} = \{(\mathbf{x}^1, \overline{\mathbf{y}}^1)(\mathbf{x}^2, \overline{\mathbf{y}}^2) \dots (\mathbf{x}^\ell, \overline{\mathbf{y}}^\ell)\}$

$\mathcal{C} := \oslash$
for $i = 1, \dots, \ell$ compute $\underline{b}_i$ and C_i
Compute $\underline{b} := \sum_i b_i$ and $\overline{C} := \sum_i C_i$
Find θ_{opt} solving (7)
Repeat
  $exit := 0$
  for $i = 1, \dots, \ell$
    Compute $\tilde{\mathbf{y}}^i := \arg\max_{\mathbf{y}} \ \theta_{opt}^T \phi(\mathbf{x}^i, \mathbf{y})$
    If $\theta_{opt}^T(\phi(\mathbf{x}^i, \overline{\mathbf{y}}^i) - \phi(\mathbf{x}^i, \tilde{\mathbf{y}}^i)) \leq 0$
      $exit := 1$
      $\mathcal{C} := \mathcal{C} \cup \{\theta^T(\phi(\mathbf{x}^i, \overline{\mathbf{y}}^i) - \phi(\mathbf{x}^i, \tilde{\mathbf{y}}^i)) \geq 0\}$
      Find θ_{opt} solving (7) s.t. $\mathcal{C}$
    end
  end
until $exit = 1$

Output: θ_{opt}

Often, real data sets do not allow a feasible solution θ. A possible way to deal with this problem is by the introduction of slack variables or relaxing the constraints by requiring the inequalities to hold subject to the small possible used-defined tolerance ε (Ricci et al., 2007). However, we argue that in such cases simply optimizing the Z-score as described earlier without adding any constraints may offer a natural and computationally attractive alternative to using soft-margin constraints.

4.3.3 RELATED WORK

It is worth noting that the Z-score has previously been used in the context of sequence alignment, although in previous work it was computed with respect to different distributions. In Doolittle (1981) Z-scores are used to assess the significance of a pairwise alignment between two aminoacid sequences and are computed calculating the mean and the standard deviation values over a random sample taken from a standard database or obtained permuting the given sequence. A high Z-score corresponds to an alignment that is less likely to occur by chance and therefore biologically significant.

To our knowledge, there are no methods to calculate the Z-scores on a set of random sequences in exact way. The only attempt to this aim is due to Booth et al. (2004). They proposed an efficient algorithm that finds the standardized score in the case of permutations of the original sequences but this approach is limited to the ungapped sequences. We have to stress that we consider a much wider range of applications (not only sequence alignment) and a slightly different definition of the Z-score: for example, for sequence alignment for each pair of given sequences the mean and standard deviation are computed over the set of all possible alignments (also with gaps and not only the optimal ones) without any permutations.

4.4 SODA: Structured Output Discriminant Analysis

Another way to extend problem (5) to the general situation of a training set $\mathcal{T}$ is to minimize the empirical risk associated to the upper bound on the relative ranking loss $\mathcal{R}^{RRU}$, defined in the usual way as:

$$\mathcal{R}_\theta^{RRU}(\mathcal{T}) \quad = \quad \sum_{i=1}^{\ell} \mathcal{L}_\theta^{RRU}(\mathbf{x}^i, \overline{\mathbf{y}}^i).$$

This simple summing of the loss for individual data points leaves the connection with FDA more intact, hence the name SODA for structured output discriminant analysis. As usual in empirical risk minimization, the hope is that minimizing the empirical risk will ensure that the expected loss $E_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\overline{\mathbf{y}})\right\}$ (here the relative ranking loss) is small as well, and we will shortly prove that this is the case in 6.1. Filling everything in, the resulting empirical risk minimization problem becomes:

$$\begin{aligned} \min_\theta \quad & \sum_{i=1}^{\ell} \frac{1}{N_i} \sum_{j=1}^{N_i} \xi_{ij}^2 \\ \text{s.t.} \quad & \theta^T(\phi(\mathbf{x}^i, \overline{\mathbf{y}}^i) - \phi(\mathbf{x}^i, \mathbf{y}_j^i)) = 1 + \xi_{ij} \quad \forall i. \end{aligned} \tag{10}$$

This is the optimization problem we solve in SODA. To solve it easily, and to elucidate more clearly the analogy with the Z-score approach, we rewrite it one more time as follows.

Proposition 6 *Optimization problem (10) is equivalent to:*

$$\max_\theta \frac{\theta^T b^*}{\sqrt{\theta^T C^* \theta}} \tag{11}$$

where we have defined $b^* = \sum_i b_i$ and $C^* = \sum_i (C_i + b_i {b_i}^T)$. Here, by equivalent we mean that the optimal values for θ differ by a constant scaling factor only. It can be solved efficiently by solving the linear system of equations $C^*\theta = b^*$.

Note that this optimization problem has the same shape as (7) and can be solved again with conjugate gradient algorithms.

Proof We can follow exactly the same procedure as in the proof of Theorem 5 to show that optimization problem (11) is equivalent with:

$$\begin{aligned} \min_\theta \sum_{i=1}^{\ell} \theta^T C_i \theta + (\theta^T b_i - 1)^2 \quad & \Leftrightarrow \quad \min_\theta \theta^T \sum_{i=1}^{\ell} (C_i + b_i b_i^T)\theta - 2\theta^T \sum_{i=1}^{\ell} b_i + 1 \\ & \Leftrightarrow \quad \min_\theta \theta^T C^* \theta - 2\theta b^* + 1. \end{aligned}$$

	MM	HMP	CRFs
Z-score	$5.67e^{-11}$	$1.97e^{-7}$	0.016
SODA	$5.12e^{-10}$	$3.13e^{-6}$	0.04

Table 1: p-values for level of noise $p = 0.4$ and for an HMM with $n_h = 2$ and $n_o = 4$.

The optimality conditions is $C^*\theta = b^*$. In a similar way as in the proof of Theorem 5, we can show that the optimality conditions of optimization problem (10) are given by $C^*\theta = \lambda b^*$, leading to the same value for θ after appropriate scaling. ■

5. Experimental Results

In this subsection we provide some experimental results for the three illustrative examples proposed: sequence labeling, sequence alignment and sequence parse learning.

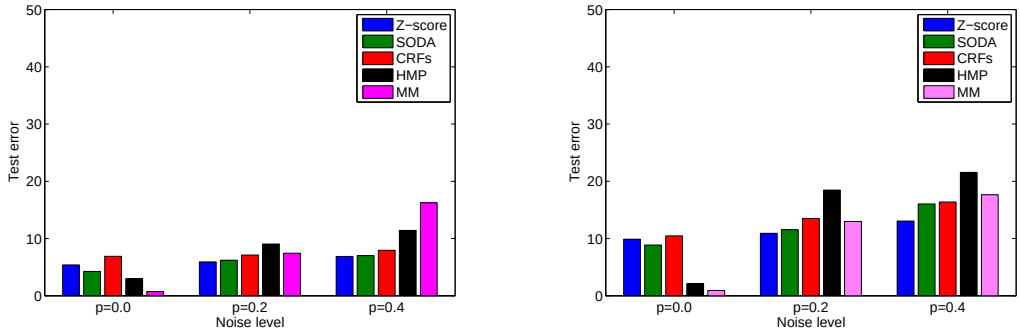
5.1 Sequence Labeling Learning

The first series of experiments, developed in the context of sequence labeling learning, analyzes the behavior of the Z-score based algorithm and of the SODA using both artificial data and sequences of text for named entity recognition. The main aim of this section is to compare our approaches with other existing DLAs on small and medium size data sets.

5.2 Simulation Results

We first present experiments that demonstrate the robustness of our approaches in problems with an increasing degree of noise. We consider two different HMMs, one with $n_h = 2$, $n_o = 4$ and one with $n_h = 3$, $n_o = 5$, with assigned transition and emission probabilities. For these models, we generate hidden and observed sequences of length 100. The training set size is fixed to 20 pairs, while the test set is made up of 100 pairs. Then we add some noise with probabilities $p \in [0,1]$ flipping labels in hidden sequences. More specifically we consider three different scenarios: absence of noise ($p = 0$), moderate level of noise ($p = 0.2$) and noisy data ($p = 0.4$). After learning the parameter, the labeling error (average number of incorrect labels) is measured. We observe the performance of the proposed approaches in comparison with other DLAs such as CRFs, hidden Markov perceptron (HMP) and a MM method with Hamming loss (SVM-struct implementation Tsochantaridis et al., 2005) and linear kernel. The regularization parameters associated to each method are determined based on the performance on a separate validation set of 100 sequences generated together with the training and the test sets. Results are averaged over 1000 training/test samples. In both cases our algorithms outperform other methods for high level of noise, as can be expected (Fig. 4). We also observe slightly better performance of the SODA with respect to the Z-score based algorithm for low p values while with the Z-score a smaller test error is achieved with very noisy data.

To assess the significance of the results obtained comparing our methods with the other DLAs we also run some paired t-tests and compute the associated p-values for both the HMM models and all the levels of noise. Here we only show the p-values obtained by the experiments with high level



(a) (b)

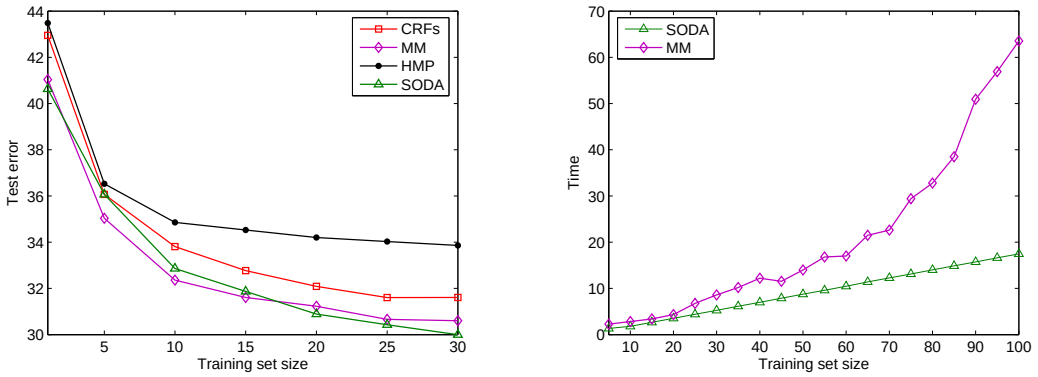
Figure 4: Average number of incorrect labels at varying level of noise for an HMM with (a) $n_h = 2$ and $n_o = 4$ and (b) $n_h = 3$ and $n_o = 5$.

	MM	**HMP**	**CRFs**
Z-score	$11.11e^{-6}$	$3.01e^{-12}$	0.0076
SODA	$8.8e^{-4}$	$5.45e^{-10}$	0.16

Table 2: p-values for level of noise $p = 0.4$ and for an HMM with $n_h = 3$ and $n_o = 5$.

of noise (Tab. 1 and Tab. 2) in order to demonstrate that our approaches significantly outperforms HMP and the MM algorithm in situations where data are noisy. In this scenario SODA and Z-score achieve better performance than CRFs even if in this case the difference of the test error is less evident. On the other hand we also observe that for separable data (absence of noise) the MM algorithm does significantly better than our algorithms (e.g., for the HMM model with $n_h = 2$ and $n_o = 4$ the p-value is $5.04e^{-9}$ for SODA and $1.06e^{-11}$ for the Z-score algorithm). A similar situation occurs also for the HMP (e.g., for the HMM model with $n_h = 2$ and $n_o = 4$ the p-value is $8.65e^{-5}$ for SODA and $4.86e^{-6}$ for the Z-score algorithm). However SODA and Z-score approach still outperform CRFs (e.g., for the HMM model with $n_h = 2$ and $n_o = 4$ the p-value is $3.51e^{-4}$ for SODA and $2.53e^{-3}$ for the Z-score algorithm).

For this series of experiments we also depict some typical learning curves computed for all DLAs considered. We show the curves associated to a HMM model with $n_h = 3$, $n_o = 5$, sequences of length equal to 50 and noise level $p = 0.2$. In this case for the MM algorithm the soft margin parameter C is set equal to 1 and a constant $\varepsilon = 10^{-12}$ specifies the accuracy for constraints to be satisfied. The maximum number of iterations of the averaged perceptron is $T = 100$. CRFs are optimized using a conjugate gradient method. Concerning our approaches we plot results just for the SODA since in this situation (moderate quantity of noise) the learning curves for the Z-score algorithm is almost superimposed to the SODA's one. For the SODA the regularization parameter is $\lambda = 10^{-8}$. The SODA performs better than other methods and among the competing DLAs, the MM approach provides the best performance (Fig. 5.a). Moreover if for the same experiment we also examine the training time we observe (Fig. 5.b) that SODA is definitely faster than the MM algorithm especially for larger data sets.



(a) (b)

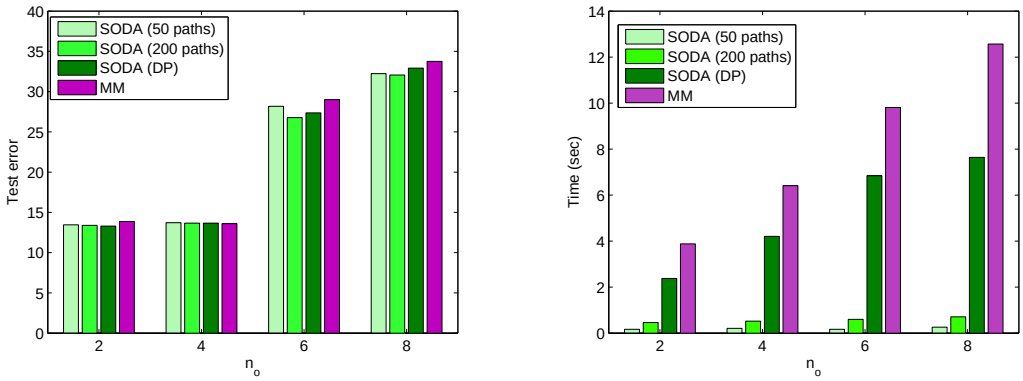
Figure 5: (a) Average number of incorrect labels and (b) computational time as function of the training set size for an HMM with $n_h = 3$ and $n_o = 5$.

A further series of experiments have been conducted to confirm the theoretical results presented in the previous subsection, that is, we want to show that learning with SODA is effectively achieved when mean and covariance matrices are estimated considering just a small subset of incorrect outputs (i.e., incorrect hidden label sequences), taken by random sampling. In fact in the situations where the size of the hidden and the observed space is large and long sequences must be considered, the computation of b^* and C^* with DP can be quite time consuming. Then using random sampling, the computational burden of DP is avoided and the labeling accuracy is still reasonably high, if a sufficient number of possible outputs is sampled. To support this claim we conduct the following experiment. Sequences of length 10 are considered. The training set is fixed to 50 pairs, the test set contains 100 pairs. Sequence pairs are generated with a level of noise $p = 0.2$ obtained by flipping labels. We pick various HMMs: the hidden alphabet size is fixed, $n_h = 3$, while n_o varies. The average labeling error on test set and the time required for computation are reported for SODA with exact matrices, when matrices are computed on a set of 50 and 200 random paths and for the MM method. Results are shown in Fig. 6. While the performance in terms of labeling error are essentially the same for all the algorithms (Fig. 6.a), the computational advantage considering the training time for the sampling approaches is considerable (Fig. 6.b).

5.3 Named Entity Recognition

The second series of experiments have been performed in the context of named entity recognition (NER). In NER phrases in text must be classified as belonging to predefined categories such as persons, organizations, locations, temporal and numerical expressions.

We consider 300 sentences extracted from the Spanish news wire article corpus used for the Special Session of CoNLL2002 on NER. Our subset contains more than 7000 tokens (about 2000 unique) and each sentence has an average length of 30 words. The hidden alphabet is limited to $n_h = 9$ different labels, since the expression types are only persons, organizations, locations and miscellaneous names. Our aim here is not to compete with large scale NER systems but to perform comparison with previous methods so we deliberately choose a small subset and an experimental



(a) (b)

Figure 6: (a) Labeling error on test set and (b) average training time as function of the observation alphabet size n_o.

setup similar to that in Altun et al. (2003b). We perform experiments into different settings: HMM features (the parameters to be determined are the transition and emission probabilities) ($\mathcal{S}_1$) and HMM features of the previous and the next words ($\mathcal{S}_2$). Experiments have been made with a 5-fold cross validation. We compare the performances of our approaches with CRFs, HMP and the MM algorithm with Hamming loss in (Altun et al., 2003b). For the SODA and the Z-score algorithm the regularization parameter is $\lambda = 10^{-8}$. For CRFs we used a public available software (Kudo, 2005) where a quasi-newton optimization technique method is used for optimization. For the MM algorithm a linear kernel is considered, $C = 1$ and $\varepsilon = 0.01$. The number of iterations of the HMP is $T = 200$.

The test errors, reported in Tab. 3, demonstrate the competitiveness of the proposed methods. SODA outperforms all the other approaches for $\mathcal{S}_1$, while it performs slightly worse than the MM algorithm for features $\mathcal{S}_2$. On the other hand for $\mathcal{S}_2$ the best performance is obtained by the Z-score algorithm.

Since the length of feature vectors is large, our approaches are generally slower than MM methods. For very large numbers of parameters, in fact, the time required to compute b^* and C^* may exceed the computation time of competing MM approaches. However, in this case, the sampling strategy can be used to approximate the matrices C^* and b^*. For example in the $\mathcal{S}_1$ setting, the average running time for the SODA is about 9967.47 sec while with SVM-struct the same task is performed in 1043.16 sec. However with the use of approximate matrices computed sampling on 150 random paths and solving the linear system by a conjugate gradient method the computational time is only 656.46 sec.

	Z-score	**SODA**	**MM**	**HMP**	**CRFs**
$\mathcal{S}_1$	11.07	*10.13*	10.97	20.99	11.96
$\mathcal{S}_2$	*7.89*	8.27	8.11	13.78	8.25

Table 3: Classification error on test set on NER (300 sentences).

	Z-score	**SODA**	**MM**	**HMP**	**CRFs**
$\mathcal{S}_1$	9.43	*8.80*	9.35	11.01	9.07
$\mathcal{S}_2$	8.57	8.01	*7.33*	7.83	8.40

Table 4: Classification error on test set on NER (1500 sentences).

To address this problem of scalability of our approach when the number of features is large we also developed a method for solving the linear systems of SODA and of the Z-score approach which is *ad hoc* for problems such as NER where the size of the observation alphabet n_o (i.e., the size of dictionary) tends to be huge while the size of the hidden alphabet n_h (i.e., the number of different labels) is moderate.

The main problem of using our algorithms for tasks with a large number of features is represented by the fact that the matrices C^* and $\bar{C}$ needs to be stored into memory. Moreover solving the corresponding linear systems with conjugate gradient techniques has computational cost $O(d^2)$ which is problematic when d is large. To overcome these difficulties we propose an approach which exploits the sparsity and the redundancy of the covariance matrices to limit the storage requirements and to solve the corresponding linear system with reduced computational cost. This approach is briefly presented in appendix B in the case of sequence labeling and HMM features but an extension of it for other possible configurations of features is possible and quite easy to derive.

Using this method we are able to perform experiments for the NER task on a large subset of the Spanish news wire article corpus of CoNLL2002. We used 1500 sentences which correspond to a dictionary of about 10000 different words. The hidden alphabet is again represented by $n_h = 9$ different labels. The experimental setting is the same of the small data set: we consider the same configuration for features ($\mathcal{S}_1$ and $\mathcal{S}_2$) and we compare the performances of our approaches with CRFs (using CRFs toolkit Kudo, 2005), HMP and the MM algorithm with Hamming loss (SVM-struct implementation Tsochantaridis et al., 2005). Experiments have been made with a 5-fold cross validation procedure and the regularization parameters which provide better performances have been set for all the methods.

From the results, shown in Tab. 4, we can draw similar conclusions that for the small data set: SODA outperforms all the other approaches for $\mathcal{S}_1$, while the MM algorithm provides the smallest test error for $\mathcal{S}_2$. It is somehow surprising that the HMP provides the second best performance for $\mathcal{S}_2$ despite its simplicity. We explain this result considering that with an increased set of features the data tend to be more separable and the HMP tend to outperform our approaches and CRFs.

Note that without having developed an *ad hoc* method such as that described in appendix B we would not have been able to run this second series of experiments on a normal machine since our $d \times d$ covariance matrices are too large to fit into memory (d is about 90000 only for set of features $\mathcal{S}_1$!). For sake of clarity we should also say that sometimes using this method we experienced numerical problems (especially for small regularization parameters) that are probably due to the fact that the approach is based on formulas for matrix inversions. Therefore in the future we plan to develop a better method (e.g., based on updating matrices decompositions such as Cholesky) in order to overcome numerical difficulties.

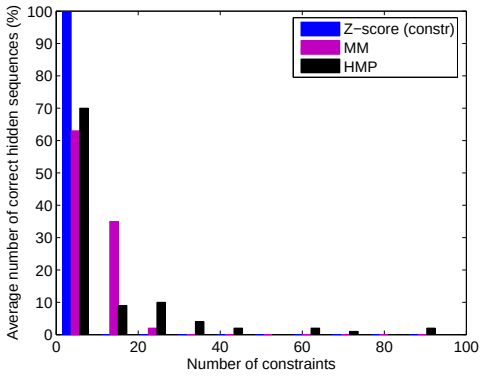


Figure 7: Average number of correctly reconstructed hidden sequences for an HMM with $n_h = 2$ and $n_o = 4$.

5.4 Z-score with Constraints

The last series of experiments shows some results associated with the Z-score approach with constraints (9). We observe experimentally that this approach improves the performance of the unconstrained problem (7) if the noise in the data is limited (i.e., in the feasible or nearly feasible case). For example for the experiments in Fig. 4 when the noise level is $p = 0$ with the constrained Z-score the labeling error is 3.96 and 5.76 respectively for the HMM with $n_h = 2$, $n_o = 4$ and for the HMM with $n_h = 3$, $n_o = 5$ while for the unconstrained problem the error is 5.39 and 9.87 respectively.

Moreover, comparing Algorithm 3 with other iterative approaches (HMP Collins 2002b and MM algorithm Tsochantaridis et al. 2005), the use of the Z-score as objective function ensures that the number of iterations is generally much smaller. Then the computational cost is greatly reduced since adding one inequality means running the Viterbi algorithm. To demonstrate this, we perform the following experiment. A pair of observed and hidden sequences of length $m = 100$ is considered. The task is to estimate the values of transition and emission probabilities such that the observed sequences are generated by the hidden one. The number of constraints needed in the training phase to reconstruct the matrices is averaged on 100 experiments. In Fig. 7 the histograms obtained binning the number of constraints needed to reconstruct the original transition and emission probabilities is shown for an HMM with $n_h = 2$ and $n_o = 4$. For sake of comparison the number of constraints needed when learning is performed with the perceptron (Collins, 2002b) and a MM approach (Tsochantaridis et al., 2005) is also provided. As expected, optimizing the Z-score, much less constraints are needed.

5.5 Sequence Alignment Learning

The second series of experiments has been performed in the context of sequence alignment learning. The aim of this section is to compare the performance of our algorithms with a traditional generative approach. Among the proposed methods we present the results associated to SODA since the performance obtained with the Z-score algorithm are nearly identical.

We construct substitution matrices with elements generated randomly but such that the values on the main diagonal are larger than the other. In particular we consider two types of matrices associated respectively with a 3 parameter model (i.e., matches, mismatches and gaps) and a 211

n	**SODA (3)**	**SODA (211)**	**Generative**	**HMP**
1	5.1 ± 1.2	96.12 ± 13.3	98.14 ± 14.5	93.8 ± 12.1
2	2.9 ± 0.8	84.7 ± 7.5	98.01 ± 12.2	83.98 ± 8.6
5	2.32 ± 1.0	74.81 ± 6.2	97.4 ± 7.4	76.13 ± 5.2
10	2.11 ± 0.7	60.08 ± 3.2	92.93 ± 5.2	57.93 ± 2.9
20	2.1 ± 0.5	43.18 ± 2.2	79.13 ± 4.2	42.68 ± 2.1
50	1.87 ± 0.3	35.56 ± 1.4	48.31 ± 2.9	31.92 ± 1.2
100	1.53 ± 0.4	30.84 ± 1.0	32.05 ± 1.5	28.4 ± 0.9
500	0.98 ± 0.3	23.47 ± 0.2	26.11 ± 0.6	21.7 ± 0.4

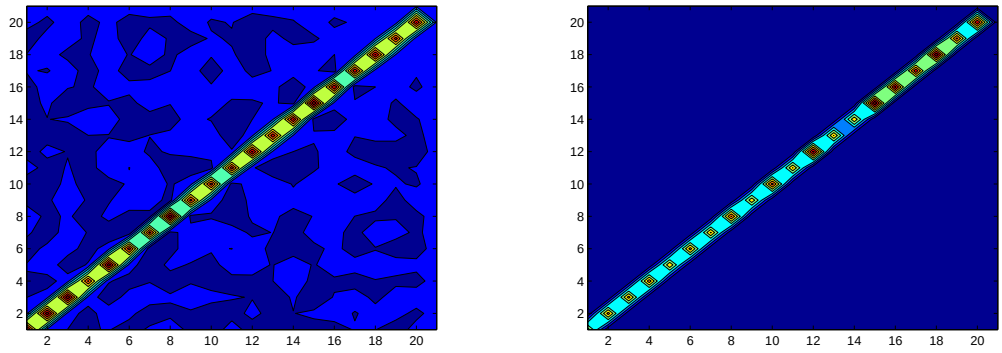
Table 5: Classification error (mean and standard deviation) on test set as function of the training set size n.

parameter models (substitution matrix plus gap penalty). Starting from these matrices we then generate random pairs of sequences of length 10 from a 20 letter alphabet. Pairs are constructed in a way that 50% of symbols between the two sequences are equal. The task we consider is to reconstruct the given matrices starting from training sets of varying size n.

Table 5 shows the results in terms of the test error (number of incorrectly aligned sequences), averaged on 100 runs. A small regularization value $\lambda = 10^{-12}$ is used for SODA. The first two columns of Tab. 5 present the test error for SODA respectively for the 3 and the 211 parameter model. As expected from theory, the convergence to zero error is faster for the 3 parameter model. For the 211 parameter model we also compare SODA with a generative sequence alignment model, where substitution rates between amino acids are computed using Laplace estimates. The gap penalty must be set manually and we choose the value $\theta_g = -0.1$ which guarantees the best performance on the test set. The third column of Tab. 5 shows the associated results: SODA performs better than the generative approach, especially for training set of small size. We also compare the performance of our method with another discriminative approach: the hidden Markov perceptron. In this situation the test error of SODA is slightly larger than that of the HMP. This is in accordance to what we observe in the sequence labeling learning task: when data are linearly separable other discriminative approaches appear more suitable than SODA.

For the SODA algorithm with the 211 parameter model and for a training set with $n = 100$ aligned pairs we also depict the substitution matrix computed by SODA and we compare it with the given one. As one can easily observe, the computed matrix has a similar structure of the correct matrix, having the elements with higher values on the diagonal (Fig. 8).

Note that, in the context of sequence alignment, being the number of parameters limited at most to 211 the training phase is not time consuming even for large training set. In fact the computational cost is dominated by the calculation of mean and covariance matrices which can be greatly sped up by sampling while solving the linear system $C^*\theta = b^*$ is indeed very fast. Here we only consider training sets of size up to 500 pairs of sequences since the advantage in terms of test error for SODA (and in general for all discriminative approaches) with respect to generative approaches is more evident for training sets of small sizes.



(a) (b)

Figure 8: Comparison between a given substitution matrix (a) and the matrix computed with SODA (b) for $n = 100$.

accession	n^o sequences	max. length
RF00032	64	27
RF00260	35	51
RF00436	24	55
RF00164	29	43
RF00480	647	52

Table 6: Summary of the data set of RNA sequences

5.6 Learning to Parse

Lastly, we analyze the RNA secondary structure prediction problem: given an RNA sequence, the task is to predict the basepairs in the sequence. With weighted context-free grammars, this prediction can be accomplished by parsing the RNA sequence. To describe basepairs in RNA sequences, we used the G6 grammar in Dowell and Eddy (2004), which we call $G = \{\Upsilon, \Sigma_x, R, S, \theta\}$, where $\Upsilon = \{S, L, F\}$, $\Sigma_x = \{\mathtt{a}, \mathtt{u}, \mathtt{g}, \mathtt{c}\}$, and $R = \{S \to LS|L, L \to \mathtt{a}F\mathtt{u}|\mathtt{u}F\mathtt{a}|\mathtt{g}F\mathtt{c}|\mathtt{c}F\mathtt{g}|\mathtt{g}F\mathtt{u}|\mathtt{u}F\mathtt{g}|\mathtt{a}|\mathtt{u}|\mathtt{c}|\mathtt{g}, F \to \mathtt{a}F\mathtt{u}|\mathtt{u}F\mathtt{a}|\mathtt{g}F\mathtt{c}|\mathtt{c}F\mathtt{g}|\mathtt{g}F\mathtt{u}|\mathtt{u}F\mathtt{g}|LS\}$. We consider RNA sequences of five families (see Tab. 6) extracted from the Rfam database (Griffiths-Jones et al., 2003). We use sequences including only standard basepairs, that is, a–u, c–g, and g–u.

Results for the experiments conducted with Z-score with constraints, with a generative model, and with the hidden Markov perceptron in a 5-fold cross validation setting are shown. Weights of the grammar are optimized with a training set, and structures associated to sequences in the test set are predicted by the Viterbi algorithm. For the Z-score with constraints, the best results obtained varying the regularization parameter are reported. For the HMP, values ranging from T=100 to T=1000 are used as the number of iterations, and the best result is shown. For the generative model, parameters are estimated with Laplace smoothing.

We measure the performance of the methods in terms of sensitivity and specificity of predicted basepairs. The sensitivity is defined as the number of correctly predicted basepairs over the number

of true basepairs, and the specificity is the number of correctly predicted basepairs over the number of predicted basepairs. In Tab. 7, the values of sensitivity and specificity corresponding to the maximum product, are shown for each algorithm. The Z-score and the HMP have comparable performances and generally outperform the generative approach. For the Z-score approach also the average number of constraints is shown: it is worth noting that only very few constraints are needed for each family, often less than the number of iterations in the HMP by an order of magnitude or more.

6. Statistical Learning Analysis

We present here two learning theory results. The first one is specific for the ranking loss in any algorithm using this hypothesis space, and hence covers the SODA algorithm. The second one applies to any algorithm using the zero-one loss with this hypothesis class, and hence covers most previous approaches.

	Z-score (constr)			**Generative**		**HMP**	
accession	sens.	spec.	n^o constraints	sens.	spec.	sens.	spec.
RF00032	100	95.98	2.0	100	95.53	100	95.59
RF00260	98.77	94.80	6.0	98.97	100	98.57	98.90
RF00436	91.11	90.61	27.6	44.16	53.30	90.27	86.53
RF00164	76.14	73.47	37.8	65.51	62.55	87.06	78.32
RF00480	99.08	89.89	78.2	99.88	86.43	98.83	94.78

Table 7: Prediction on 5-fold cross validation. Average sensitivity and specificity are shown.

6.1 Rademacher Theory for SODA

Here we present a Rademacher bound for the SODA showing that learning based on this upper bound on the relative rank is effectively achieved. For full generality, we want our bound to hold also in the case where the matrices μ and C are estimated by sampling, as we suggested in subsection 3.2.2. We also provide some experimental results for this in the following subsection. Hence, our bound needs to account for finiteness in two ways. First of all for each input-output pair only a limited number n of incorrect outputs may be considered to estimate μ and C; secondly only a finite number ℓ of input-output pairs is given in the training set.

In appendix C, we prove the following theorem. It shows that the empirical expectation of the estimated loss (estimated by computing C and b by random sampling) is a good approximate upper bound for the expected loss. Hereby it is good to keep in mind that this loss itself is an upper bound for the relative ranking loss, such that the Rademacher bound is also a bound on the expectation of the relative ranking loss.

Theorem 7 (Rademacher bound for SODA) *With probability at least* $1-\delta$ *over the joint of the random sample* $\mathcal{T}$ *and the random samples from the output space for each* $(\boldsymbol{x},\bar{\boldsymbol{y}}) \in \mathcal{T}$ *that are taken to approximate the matrices* C^* *and* b^**, the following bound holds for any* θ *with squared norm smaller than* c*:*

$$E_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\} \quad \leq \quad \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\widehat{\mathcal{L}}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\}$$

$$+ \quad \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}\right\} + \hat{\mathfrak{R}}_2$$
$$+ \quad 3M\sqrt{\frac{\log(2\ell/\delta)}{2n}} + 3M\sqrt{\frac{\log(2/\delta)}{2\ell}}.$$

whereby we assume that the number of random samples for each training pair is equal to n.

The Rademacher complexity terms $\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}$ and $\hat{\mathfrak{R}}_2$ decrease with $\frac{1}{\sqrt{n}}$ and $\frac{1}{\sqrt{\ell}}$ respectively, such that the bound becomes tight for increasing n and ℓ, as long as n grows faster than $\log(\ell)$.

For details relating to the exact value of the Rademacher complexity terms, the value of the constant M, and the proofs, we refer to the appendix C.

6.2 PAC Bound

In appendix D, we prove the following theorem that applies to a generic DLA: given a training set of sample pairs $(\mathbf{x}^i, \overline{\mathbf{y}}^i)$, can we learn to predict the output for a previously unseen observation? For example, given a training set of aligned protein sequences, can we learn how to align a previously unseen pair? Or, given a training set of correctly parsed sentences, can we learn how to parse a previously unseen sentence? To be clear, in this section we consider the zero-one loss only, which has been considered most often in previous work on structured output learning.

DLAs directly learn the model parameters such that the accuracy of the prediction is somehow optimized. All these algorithms are in some sense empirical risk minimizers, in that they optimize the prediction performance on a training set. However till now there have been few works trying to address the question whether a small empirical risk guarantees a small expected risk. A first generalization bound has been developed by Collins for the case of the perceptron algorithm (Collins, 2002a) and a capacity bound in terms of covering numbers for the maximum margin approach has been proposed by Taskar et al. (2003). These bounds have subsequently been reconsidered in McAllester (2007) and have been improved in order to achieve consistency for any arbitrary loss. In this paper we answer the learnability question affirmatively from another point of view, independent of the learning approach taken, and we propose a new PAC bound which makes use of a result which bounds the cardinality of the hypothesis space of prediction functions derived from DLAs.

We go back to the original problem of structured output learning. Given is a training set $\mathcal{T} = \{(\mathbf{x}^1, \overline{\mathbf{y}}^1), (\mathbf{x}^2, \overline{\mathbf{y}}^2), \ldots, (\mathbf{x}^\ell, \overline{\mathbf{y}}^\ell)\}$ of observation-output pairs, with observations $\mathbf{x}^i \in \mathcal{X}$ and outputs $\overline{\mathbf{y}}^i \in \mathcal{Y}$ jointly drawn i.i.d. from an unspecified probability distribution $P(\mathbf{x}, \mathbf{y})$. Based on $\mathcal{T}$ we want to infer a prediction function $h_\theta : \mathcal{X} \to \mathcal{Y}$ such that the probability $P(h_\theta(\mathbf{x}) = \overline{\mathbf{y}})$ of an observation-output pair $(\mathbf{x}, \overline{\mathbf{y}})$ with $h_\theta(\mathbf{x}) = \overline{\mathbf{y}}$ is as large as possible. For learnability, the choice of h_θ should be restricted to a limited hypothesis space, and DLAs provide one way to achieve this.

Our hypothesis space $\mathcal{H}$ is the space containing all prediction functions h_θ with $\theta \in \mathbb{R}^d$, defined as:

$$h_\theta(\mathbf{x}) \quad = \quad \arg\max_{\mathbf{y}\in\mathcal{Y}} \; s_\theta(\mathbf{x}, \mathbf{y}) = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; \theta^T \phi(\mathbf{x}, \mathbf{y}),$$

and this for a fixed feature map ϕ with integer features between 0 and C.

For this hypothesis space we can prove (in appendix D) the following theorem.

Theorem 8 (On the PAC-learnability of structured output prediction) *Given a hypothesis space $\mathcal{H}$ of prediction functions h_θ as defined above. Furthermore, consider a training set $\mathcal{T} =$*

$\{(\boldsymbol{x}^1,\bar{\boldsymbol{y}}^1),(\boldsymbol{x}^2,\bar{\boldsymbol{y}}^2),\ldots,(\boldsymbol{x}^\ell,\bar{\boldsymbol{y}}^\ell)\}$ of observation-output pairs, sampled i.i.d. from a fixed but unknown distribution. Then, with probability at least $1-\delta$ over the random sample $\mathcal{T}$, for any $h_\theta \in \mathcal{H}$ for which $h_\theta(\boldsymbol{x}^i)=\bar{\boldsymbol{y}}^i$ for all $(\boldsymbol{x}^i,\bar{\boldsymbol{y}}^i)\in\mathcal{T}$ the expected risk can be bounded as:

$$E_{(\boldsymbol{x},\bar{\boldsymbol{y}})\sim\mathcal{D}}\{h_\theta(\boldsymbol{x})\neq\bar{\boldsymbol{y}}\}\leq\frac{d^2\log(2C)-(d-2)\log(d-2)+d-\log(\delta)}{\ell}.$$

We can thus conclude that learning is guaranteed as soon as $\ell >> d^2\log(2C)$.

This result proves that learning based on DLAs can be achieved effectively, and d and C are the factors that are relevant in determining the learning rate. Importantly this bound holds regardless of the method used to estimate the parameter vector θ. Interestingly, it suggests that the number C (bounded by to the number of cliques for DLAs derived from PGMs) is less important than the number of parameters d.

Note that this bound only holds for the realizable case, such that its practical relevance is limited. Furthermore, unlike the results from McAllester (2007), the bound does not depend on the norm of the weight vector θ, making it loose for small values of $\|\theta\|$. Nevertheless, we believe it is of interest due to the simplicity of its derivation based on results from combinatorics and basic PAC theory.

7. Conclusions

We have presented a formal framework for learning to predict over structured output spaces. The hypothesis space we consider is based on linear scoring functions, much like most previous approaches to this problem.

The distribution of this linear scoring function over all possible outputs contains information that we can use to train the parameters of the learning algorithm. We can compute efficiently the first two moments of this distribution, and we use them to derive convex objective functions for parameter optimization.

In this way, we have derived two new efficient algorithms for structured output prediction that rely on these statistics, both of which can be solved by solving one linear system of equations.

Interestingly, and thanks to the use of the moments, one of the proposed objective functions (SODA) represents a convex upper bound on the relative ranking loss: the fraction of outputs from the output space that rank better than the correct output. Thanks to this property, SODA naturally and adequately deals with the infeasible case where there exists no parameter setting for which the correct given pairs are optimal. We justify this fact theoretically, providing a Rademacher bound, and experimentally, reporting results that are competitive with existing methods, and better than other methods in the infeasible case.

Acknowledgments

We are most grateful to Nobuhisa Ueda since without him the section on SCFG's would not have been possible. We also thank the anonymous reviewers for providing us with their valuable comments. This work was partially supported by NIH grant R33HG00 3070-01, the EU project SMART and the PASCAL network of excellence. The work of Nello Cristianini is partially supported by a Royal Society Wolfson Merit Award.

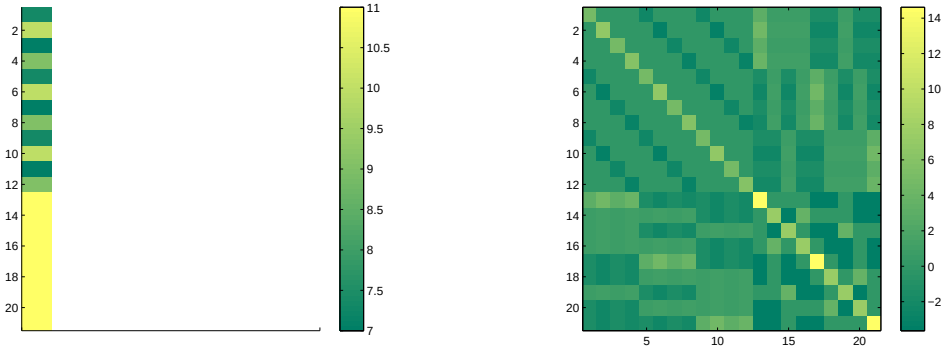
Appendix A. Proof of Proposition 3

The number of DP routines needed to compute μ and C are $7n_o+6$.

In fact in general in the mean vector μ there are n_o+1 different values. All the elements associated to transition probabilities assume the same values while for emission probability $\mu^e_{pq}=\mu^e_{ef}$, $\forall\, q=f$.

We analyze the structure of the matrix C. It is a symmetric block matrix made basically by three components: the block associated to emission probabilities, that of transition probabilities and that relative to mixed terms. To compute it $6n_o+5$ DP routines are required. In the emission part there are $2n_o$ possible different values since $c^e_{pq}=c^e_{ef}$, $\forall q=f$, $c^e_{pqp'q'}=0$, $\forall q\neq q'$ and $c^e_{pqp'q'}=c^e_{efe'f'}$ $\forall q=q'=f=f'$. In the transition block there are only 5 possible different values. In particular for the second order moments, it holds that $c^t_{pz}=c^t_{eg}$, $\forall p=z=e=g$ and $c^t_{pz}=c^t_{eg}$, $\forall p=e$, $z=g$ and $p\neq z$. For the remaining three values there holds that $c^t_{pzp'z'}=0$, $\forall p\neq p'$, $z\neq z'$, $c^t_{pzp'z'}=c^t_{ege'g'}$, $\forall p=p'$, $z\neq z'$, $e=e'$, $g\neq g'$ and $c^t_{pzp'z'}=c^t_{ege'g'}$, $\forall p\neq p'$, $z=z'$, $e\neq e'$, $g=g'$. The block relative to mixed terms is made of $4n_o$ possible different value. In fact there are n_o values $c^{et}_{pqp'z}$ with $p=p'=z'$, n_o values $c^{et}_{pqp'z}$, with $p=p'$, $p'\neq z'$, n_o values $c^{et}_{pqp'z}$, with $p=z'$, $p'\neq z'$ and n_o values $c^{et}_{pqp'z}$, with $p\neq p'$, $z\neq z'$.

The redundancy in the structure of matrix C and of the vector μ can be observed in Fig. 9 for an HMM with $n_s=3$ and $n_o=4$.



(a) (b)

Figure 9: Mean vector and covariance matrix for an HMM with $n_s=3$ and $n_o=4$.

Appendix B. Solving Linear Systems for Large Feature Spaces

This paragraph provides a brief description of an approach for solving the linear systems $C^*\theta=b^*$ and $\bar{C}\theta=\bar{b}$ avoiding to store the entire matrices C^* and $\bar{C}$. This approach is suited to sequence labeling problems and HMM features and it is particularly effective for problems when n_h, the size of the hidden state alphabet, is small and n_o, the size of the observation alphabet, is large.

We describe the procedure to solve $C^*\theta=b^*$. In fact it subsumes the method for solving $\bar{C}\theta=\bar{b}$. The main idea behind this procedure is that exploiting the structure of C^* we can store just a part of it and compute the optimal parameter vector θ effectively.

The matrix C^* in case of sequence labeling and HMMs features is sparse and redundant. In fact this matrix is given by the sum of two parts: $\bar{C}=\sum_i C_i$ and $BB^T=\sum_i b_i b_i^T$. We first consider the

first part $\bar{C} = \sum_i C_i$. Each C_i is very sparse and has a regular structure as discussed in appendix A. Then also the matrix $\bar{C}$ has the same structure, that is, is a matrix made by four block:

$$\bar{C} = \begin{pmatrix} E & M \\ M^T & T \end{pmatrix}.$$

Here E denotes the block associated to emission probabilities, T that corresponding to transition probabilities and M that relative to mixed terms. We are interested in finding the inverse of the matrix $\bar{C}$ without storing it entirely. Note that in many situations (e.g., sequence labeling problems for text analysis such as NER or POS) the emission part represents the main bottleneck in the computation of the inverse since its size is dependent on n_o (e.g., the size of the dictionary). The size of the transition part instead is usually moderate since it is given by n_h^2 (e.g., the number of different tags). The inverse of $\bar{C}$ can be computed by:

$$\bar{C}^{-1} = \begin{pmatrix} E^{-1} + E^{-1}MP^{-1}M^TE^{-1} & -E^{-1}MP^{-1} \\ -P^{-1}M^TE^{-1} & P^{-1} \end{pmatrix}$$

where $P = T - M^TE^{-1}M$ represent the Schur complement of E. The inverse of the matrix E can be computed easily due to the structure of the matrix E. In fact E is a block matrix:

$$E = \begin{pmatrix} E_d & E_o & E_o & \cdots & E_o \\ E_o & E_d & E_o & \cdots & E_o \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ E_o & \cdots & E_o & E_o & E_d \end{pmatrix}$$

where E_d and E_o are both diagonal matrices. Therefore we can rewrite the matrix E as:

$$\begin{aligned} E = & \begin{pmatrix} E_d - E_o & 0 & 0 & \cdots & 0 \\ 0 & E_d - E_o & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & E_d - E_o \end{pmatrix} + \begin{pmatrix} I \\ I \\ \vdots \\ I \end{pmatrix} E_o \begin{pmatrix} I & I & \cdots & I \end{pmatrix} \\ = & \quad D + H^T E_o H. \end{aligned}$$

Then the inverse can be computed easily considering the formula for the inverse of a sum of matrices:

$$E^{-1} = D^{-1} - D^{-1}H^T(I + E_oHD^{-1}H^T)^{-1}E_oHD^{-1}$$

where D is a diagonal matrix. Due to the special structure of D, H and E, it turns out that the inverse of E is also a block matrix with similar structure of E, that is,

$$E^{-1} = \begin{pmatrix} \bar{E}_d & \bar{E}_o & \bar{E}_o & \cdots & \bar{E}_o \\ \bar{E}_o & \bar{E}_d & \bar{E}_o & \cdots & \bar{E}_o \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \bar{E}_o & \cdots & \bar{E}_o & \bar{E}_o & \bar{E}_d \end{pmatrix}$$

where $\bar{E}_d = (E_d - E_o)^{-1}$ and $\bar{E}_o = \bar{E}_d(I + E_oN_H\bar{E}_d)^{-1}\bar{E}_d$ (N_H is a diagonal matrix with elements on the main diagonal equal to n_h). Then it is not necessary to compute and store the entire matrix E^{-1} but only the small blocks $\bar{E}_d$ and $\bar{E}_o$.

Once the matrix E^{-1} has been obtained then the computation of the Schur complement P and its inverse it is straightforward. This is not a time consuming procedure since its size n_h^2 is typically small and E^{-1} is very sparse. Due to the redundancy and the particular structure of the matrix M we can also compute quite easily all the other terms. In particular the matrix obtained by $E^{-1}MP^{-1}M^TE^{-1}$ is a block matrix made by $n_h \times n_h$ equal blocks. Then it suffices to compute and to store just one of each block.

The inverse of the matrix $\bar{C}$ has then been obtained and we can use it directly to compute the solution of the linear system for the Z-score approach $\theta = \bar{C}^{-1}\bar{b}$. Instead if we want to obtain the optimal parameter vector associated to SODA we must compute the solution of the linear system $(\bar{C}+BB^T)\theta = b^*$. In practice what we need is a method to perform n rank one updates (one for each sample in the training set) of the inverse of the matrix $\bar{C}$ without storing the matrices $\bar{C}^{-1}$ and BB^T entirely. We can use the Sherman-Morrison-Woodbury formula:

$$\bar{C}+BB^T = \bar{C}^{-1} - \bar{C}^{-1}B(I+B^T\bar{C}^{-1}B)B^T\bar{C}^{-1}$$

to calculate the solution of our linear system:

$$\theta = (\bar{C}+BB^T)^{-1}b^* = \bar{C}^{-1}b^* - \bar{C}^{-1}B(I+B^T\bar{C}^{-1}B)B^T\bar{C}^{-1}b^*.$$

In practice we first compute $z = \bar{C}^{-1}b^*$ and use this value to solve the linear system by Cholesky decomposition:

$$(I+B^T\bar{C}^{-1}B)t = B^Tz.$$

Note that the cost of this operation is $O(n^3)$ but it is usually moderate since $n << d$. The computational cost here is dominated by the calculation of z since in theory it requires d^3 multiplications. However in practice this cost is still reasonable since $\bar{C}^{-1}$ tend to be sparse. Once we have computed t we can obtain our solution for SODA simply by $\theta = z - \bar{C}^{-1}Bt$.

Appendix C. Proof of the Rademacher Bound

We consider two types of randomness in our bound: the randomness in choosing the finite sample of training input-output pairs, and the randomness in sampling from the output space for each of the training inputs. Our aim is to provide a learning theory bound for the expected relative rank of the score $s_\theta(\mathbf{x},\bar{\mathbf{y}})$ among all scores $s_\theta(\mathbf{x},\mathbf{y})$ for all $\mathbf{y} \in \mathcal{Y}$.

More exactly, we are interested in bounding $E_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\}$ where the value of the loss $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) = E_{\mathbf{y}}\left\{\left[\theta^T\left(\phi(\mathbf{x},\bar{\mathbf{y}})-\phi(\mathbf{x},\mathbf{y})\right)-1\right]^2\right\}$ is known to be an upper bound on the relative rank of the score of $(\mathbf{x},\bar{\mathbf{y}})$ among all scores of $(\mathbf{x},\mathbf{y})$ for all possible $\mathbf{y}$ (see Theorem 5).

For clarity, let us first consider a fixed training pair $(\mathbf{x},\bar{\mathbf{y}})$. We will derive a Rademacher bound that shows that the loss function $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})$ is approximately upper bounded by its empirical estimate $\widehat{\mathcal{L}}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) = \widehat{E}_{\mathbf{y}}\left\{\left[\theta^T\left(\phi(\mathbf{x},\bar{\mathbf{y}})-\phi(\mathbf{x},\mathbf{y})\right)-1\right]^2\right\}$, obtained by averaging over a random sample of n values of $\mathbf{y}$. In particular we will show that with a probability of at least $1-\delta_1$ over the random sample of size n:

$$\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) \leq \widehat{\mathcal{L}}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) + \hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})} + 3M\sqrt{\frac{\log(1/\delta_1)}{2n}},$$

with $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$ an empirical Rademacher complexity term. The constant M is an upper bound on the value of $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})$ valid for all allowable θ, and is a finite number. Such an upper bound can be computed as $M = (C\sqrt{dc}+1)^2$, by considering the constraint $\|\theta\|^2 \leq c$ and the fact that for all d features $0 \leq \phi_i(\mathbf{x},\mathbf{y}) \leq C$.

Second, we will show that the expectation of $\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})$ over $(\mathbf{x},\bar{\mathbf{y}})$ is approximately upper bounded by its empirical expectation over the training set of size l. We will show that with probability at least $1-\delta_2$ over the training set $\mathcal{T}$ of size ℓ,

$$E_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} \leq \widehat{E}_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} + \hat{\mathfrak{R}}_2 + 3M\sqrt{\frac{\log(1/\delta_2)}{2\ell}},$$

with $\hat{\mathfrak{R}}_2(\mathcal{T})$ an empirical Rademacher complexity term, and with the same constant M.

Putting these two partial results together with $\delta_1 = \frac{\delta}{2\ell}$ and $\delta_2 = \frac{\delta}{2}$, we have shown the following theorem:

Theorem 9 (Rademacher bound for SODA) *With probability at least $1-\delta_2-\ell\delta_1 = 1-\delta$ over the joint of the random sample $\mathcal{T}$ and the random samples from the output space for each $(\boldsymbol{x},\bar{\boldsymbol{y}}) \in \mathcal{T}$, the following bound holds for any θ with squared norm smaller than c:*

$$\begin{aligned} E_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\mathcal{L}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\} &\leq \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\widehat{\mathcal{L}}_\theta^{RRU}(\boldsymbol{x},\bar{\boldsymbol{y}})\right\} \\ &+ \widehat{E}_{(\boldsymbol{x},\bar{\boldsymbol{y}})}\left\{\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}\right\} + \hat{\mathfrak{R}}_2 \\ &+ 3M\sqrt{\frac{\log(2\ell/\delta)}{2n}} + 3M\sqrt{\frac{\log(2/\delta)}{2\ell}}. \end{aligned}$$

The first term on the right hand side of the inequality is the empirical risk, which is minimized on the training set. The next two terms are Rademacher complexity terms, and we will see below that these decrease to zero with increasing ℓ and n. Also the last two terms decrease to zero with increasing ℓ and n, as long as n is chosen to increase faster than $\log(\ell)$.

Both these partial results can be derived by using the generalization error bound in Bartlett and Mendelson (2002, Theorem 2) and applying the McDiarmid's concentration inequality (McDiarmid, 1989). In the following we show how to compute upper bounds on the empirical Rademacher complexities $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$ and $\hat{\mathfrak{R}}_2$.

C.1 Rademacher Bound for the Relative Rank of a Single Pair

Given a training pair $(\mathbf{x},\bar{\mathbf{y}})$ and a set $\{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_n\}$ of n randomly sampled values for $\mathbf{y}$ corresponding to $\mathbf{x}$. For notational convenience, let us denote $\varphi_j = \phi(\mathbf{x},\bar{\mathbf{y}}) - \phi(\mathbf{x},\mathbf{y}_j)$. Then we can write the empirical estimate of the loss as

$$\widehat{\mathcal{L}}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}}) = \frac{1}{n}\sum_{j=1}^{n}\left(\theta^T\varphi_j - 1\right)^2.$$

Using this notation, and with σ a vector of length n containing the independently distributed Rademacher variables σ_j being uniformly distributed over 1 and -1, the Rademacher complexity term $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$ can be written and bounded as:

$$\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})} = E_\sigma\left\{\max_\theta\left|\frac{2}{n}\sum_{j=1}^{n}\sigma_j\left(\theta^T\varphi_j - 1\right)^2\right|\right\}$$

$$
\begin{aligned}
&= E_\sigma\left\{\max_\theta\left|\frac{2}{n}\sum_{j=1}^{n}\sigma_j\left((\theta^T\varphi_j)^2-2(\theta^T\varphi_j)+1\right)\right|\right\}\\
&\leq E_\sigma\left\{\max_\theta\frac{2}{n}\left(\left|\sum_{j=1}^{n}\sigma_j(\theta^T\varphi_j)^2\right|+\left|\sum_{j=1}^{n}2\sigma_j(\theta^T\varphi_j)\right|+\left|\sum_{j=1}^{n}\sigma_j\right|\right)\right\}. \qquad (12)
\end{aligned}
$$

The first equality is the definition of the empirical Rademacher complexity, and the second equality is a trivial rewriting. The first inequality holds since the absolute value of the sum is smaller than or equal to the sum of absolute values. We now first use the fact that the maximum of a sum of functions is smaller than or equal to the sum of their maxima to show that:

$$
\begin{aligned}
(12) \quad &\leq \frac{2}{n}E_\sigma\left\{\max_\theta\left|\sum_{j=1}^{n}\sigma_j(\theta^T\varphi_j)^2\right|+\max_\theta\left|\sum_{j=1}^{n}2\sigma_j(\theta^T\varphi_j)\right|+\max_\theta\left|\sum_{j=1}^{n}\sigma_j\right|\right\}\\
&= \frac{2}{n}E_\sigma\left\{\sqrt{\max_\theta\left(\sum_{j=1}^{n}\sigma_j(\theta^T\varphi_j)^2\right)^2}\right\}\\
&\quad +\frac{2}{n}E_\sigma\left\{\sqrt{\max_\theta\left(\sum_{j=1}^{n}2\sigma_j(\theta^T\varphi_j)\right)^2}\right\}+\frac{2}{n}E_\sigma\left\{\sqrt{\max_\theta\left(\sum_{j=1}^{n}\sigma_j\right)^2}\right\}. \qquad (13)
\end{aligned}
$$

Here we used the fact that the absolute value is the square root of the square, and that the square root of a positive function is maximized when that function itself is maximized. We proceed by rewriting this expression using bracket notation, $\langle \mathbf{a}, \mathbf{b}\rangle$ denoting the inner product between vectors (or matrices) **a** and **b**. Furthermore, we use the fact that the maximum of a sum (or expectation) is smaller than or equal to the sum (or expectation) of the maxima of the individual sums, to show that:

$$
\begin{aligned}
(13) \quad &\leq \frac{2}{n}\sqrt{E_\sigma\left\{\max_\theta\sum_{j,k=1}^{n}\sigma_j\sigma_k\langle\theta\theta^T,\varphi_j\varphi_j^T\rangle\langle\theta\theta^T,\varphi_k\varphi_k^T\rangle\right\}}\\
&\quad +\frac{2}{n}\sqrt{E_\sigma\left\{\max_\theta\sum_{j,k=1}^{n}4\sigma_j\sigma_k\langle\theta,\varphi_j\rangle\langle\theta,\varphi_k\rangle\right\}}+\frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}\sigma_j\sigma_k\right\}}. \qquad (14)
\end{aligned}
$$

We now invoke the Cauchy-Schwartz inequality, and use the fact that $\|\theta\|^2 \leq c$ and hence $\|\theta\theta^T\|^2 \leq c^2$, to show that:

$$
\begin{aligned}
(14) \quad &\leq \frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}\sigma_j\sigma_k c^2\|\varphi_j\|^2\|\varphi_k\|^2\right\}}\\
&\quad +\frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}4c\sigma_j\sigma_k\|\varphi_j\|\|\varphi_k\|\right\}}+\frac{2}{n}\sqrt{E_\sigma\left\{\sum_{j,k=1}^{n}\sigma_j\sigma_k\right\}}. \qquad (15)
\end{aligned}
$$

Since for $i \neq k$, there holds that $E_\sigma\left\{\sigma_j\sigma_k\right\} = 0$ and $E_\sigma\left\{\sigma_j^2\right\} = 1$, we can finally write that:

$$(15) \quad = \quad \frac{2}{\sqrt{n}}\left(c\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^4} + 2\sqrt{c}\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^2} + 1\right).$$

In summary, we have found the following upper bound on the first empirical Rademacher complexity:

Proposition 10 (Rademacher complexity $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$) *The Rademacher complexity term $\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})}$ can be bounded as:*

$$\hat{\mathfrak{R}}_{1,(\boldsymbol{x},\bar{\boldsymbol{y}})} \leq \frac{2}{\sqrt{n}}\left(c\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^4} + 2\sqrt{c}\sqrt{\frac{1}{n}\sum_{j=1}^{n}\|\varphi_j\|^2} + 1\right),$$

which, given the boundedness of $\|\varphi_j\|$, decreases to zero as n increases to infinity, as required.

C.2 Rademacher Complexity for the Empirical Expectation of the Loss

Given a randomly sampled training set $\mathcal{T} = \{(\mathbf{x}^1,\bar{\mathbf{y}}^1),(\mathbf{x}^2,\bar{\mathbf{y}}^2),\ldots,(\mathbf{x}^\ell,\bar{\mathbf{y}}^\ell)\}$. The empirical expectation of the loss can be written as:

$$\begin{aligned}\widehat{E}_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} &= \frac{1}{\ell}\sum_{i=1}^{\ell}\mathcal{L}_\theta^{RRU}(\mathbf{x}^i,\bar{\mathbf{y}}^i)\\ &= \frac{1}{\ell}\sum_{i=1}^{\ell}\left(\frac{1}{N_i}\sum_{j=1}^{N_i}(\theta^T\varphi_j^i - 1)^2\right),\end{aligned}$$

where $\varphi_j^i = \phi(\mathbf{x}^i,\mathbf{y}_j^i) - \phi(\mathbf{x}^i,\bar{\mathbf{y}}^i)$ and N_i is the cardinality of the output space corresponding to $\mathbf{x}^i$.

For notational convenience, let us introduce the matrix Φ^i containing all vectors ${\varphi_j^i}^T$ as its rows. Then we can rewrite the expected loss function in a more compact form as:

$$\begin{aligned}\widehat{E}_{(\mathbf{x},\bar{\mathbf{y}})}\left\{\mathcal{L}_\theta^{RRU}(\mathbf{x},\bar{\mathbf{y}})\right\} &= \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{\|\Phi^i\theta - \mathbf{1}\|^2}{N_i}\\ &= \frac{1}{\ell}\sum_{i=1}^{\ell}\frac{\langle\theta\theta^T,{\Phi^i}^T\Phi^i\rangle - 2\langle\theta,{\Phi^i}^T\mathbf{1}\rangle + \langle\mathbf{1},\mathbf{1}\rangle}{N_i}.\end{aligned}$$

We have rewritten this in a form that contains a term linear in $\theta\theta^T$, a term linear in θ, and a constant term. It is exactly this decomposition of the empirical expectation of the loss that has allowed us to derive a bound on the Rademacher term $\hat{\mathfrak{R}}_{1,(\mathbf{x},\bar{\mathbf{y}})}$, so we can follow the same principles here. We omit the details here, and just state the result:

Proposition 11 (Rademacher complexity $\hat{\mathfrak{R}}_2$) *The Rademacher complexity term $\hat{\mathfrak{R}}_2$ can be bounded as:*

$$\hat{\mathfrak{R}}_2 \leq \frac{2}{\sqrt{\ell}}\left(c\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}\left(\frac{\sum_{j,k=1}^{N_i}({\varphi_j^i}^T\varphi_k^i)^2}{N_i^2}\right)} + 2\sqrt{c}\sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell}\left\|\frac{\sum_{j=1}^{N_i}\varphi_j^i}{N_i}\right\|^2} + 1\right),$$

which again decreases to zero as ℓ increases to infinity.

Appendix D. Proof of the PAC Bound

This section contains the proof of the PAC bound stated in subsection 6.2.

D.1 Bounding the Effective Cardinality of the Hypothesis Space

The number of possible functions mapping the input space on the output space is potentially huge: $|\mathcal{Y}|^{|\mathcal{X}|}$ for an observation space of size $|\mathcal{X}|$ and an output space of size $|\mathcal{Y}|$. To make this more concrete, for the HMM prediction problem discussed earlier, this is equal to $\left(n_h^m\right)^{n_o^m} = n_h^{mn_o^m}$, which is doubly exponential in the length of the sequences m.

It would clearly be impossible to achieve learning if we had to consider all of these possible functions mapping observations onto outputs. However, we will show that the hypothesis class of prediction functions defined above contains only a very small subset of these functions. This means that, while the cardinality of functions h_θ is infinite (one such function for each $\theta \in \mathbb{R}^d$), the effective cardinality is low, since many of these functions are equivalent. We will subsequently use this upper bound on the effective cardinality to obtain a PAC bound on the generalization.

To upper bound the effective cardinality of the hypothesis space $\mathcal{H}$, we borrow and reformulate the so-called *few inference functions theorem* by Elizalde (to appear) in the terminology of the present paper:

Theorem 12 (Elizalde) *Let d and C be fixed positive integers. Let $\phi : \mathcal{X} \times \mathcal{Y} \to \{0,1,\ldots,C\}^d$ be a fixed function (called the feature map). Then the hypothesis space $\mathcal{H}$ defined as $\mathcal{H} = \left\{h_\theta | h_\theta(\boldsymbol{x},\boldsymbol{y}) = \arg\max_{\boldsymbol{y}} \theta^T \phi(\boldsymbol{x},\boldsymbol{y}) | \theta \in \mathbb{R}^d\right\}$ has an effective cardinality of at most*

$$K = \frac{2^{d^2-d+1}}{(d-2)!} C^{d(d-1)}.$$

that is, the number of different *prediction functions in $\mathcal{H}$ is at most K.*

D.2 A PAC Bound for Learning Prediction Functions

Based on the effective cardinality of the hypothesis space we can now derive a PAC bound on the expected risk. Let us derive the bound for the case where the empirical risk is equal to zero. Deriving a PAC bound in the case of nonzero empirical risk is a well-known variation (Vapnik, 1998), and we will not discuss it in the current paper.

The probability of classifying all the ℓ training examples correctly with any fixed prediction function is:

$$P(\text{all } \ell \text{ correct } | p) \leq (1-p)^\ell \leq \exp(-\ell p)$$

where p is the expected risk for this prediction function. However in general, the prediction function is chosen from the hypothesis space $\mathcal{H}$. The probability over the sample that *any of K prediction functions* with an expected error rate of at least p faultlessly performs on all training examples is bounded by

$$P(\text{all } \ell \text{ correct, for any prediction function } \in \mathcal{H} | p) \quad \leq \quad K \exp(-\ell p),$$

where K is the effective cardinality of $\mathcal{H}$. Hence, the probability to get a zero training set error, for any of the prediction functions and thus for any of the parameter values, is at most $K \exp(-\ell p)$,

where p is the minimal error probability. Thus, we found an upper bound which holds with confidence at least δ for the expected error rate p as:

$$p \leq \frac{\log(K) - \log(\delta)}{\ell}$$

As we have seen in Theorem 12, the effective cardinality of $\mathcal{H}$ is upper bounded by $K = \frac{2^{d^2-d+1}}{(d-2)!} C^{d(d-1)}$, and thus (using $\log(n!) \leq n\log(n) - n$), we have proven the Theorem 8.

Appendix E. Algorithms

This section contains additional formulas that can be used for moments computation respectively in case of sequence labeling (Algorithm 4) and of sequence alignment (Algorithm 5).

Algorithm 4 Extra formulas for sequence labeling

11: **if** $z = i$ **then** $M := 1$
12: $\mu^t_{pz}(i,j) := \frac{\sum_i \mu^t_{pz}(i,j-1)\pi(i,j-1) + M\pi(p,j-1)}{\pi(i,j)}$

5: **if** $q = x_1 \wedge p = i$ **then** $v^e_{pq}(i,1) := 1$
11: **if** $q = x_j \wedge p = i$ **then** $M := 1$
12: $v^e_{pq}(i,j) := \frac{\sum_i (v^e_{pq}(i,j-1) + 2M\mu^e_{pq}(i,j-1) + M)\pi(i,j-1)}{\pi(i,j)}$

11: **if** $q' = x_j \wedge p' = i$ **then** $M_1 := 1$
if $q = x_j \wedge p = i$ **then** $M_2 := 1$
12: $v^e_{pqp'q'}(i,j) := \frac{\sum_i (v^e_{pqp'q'}(i,j-1) + M_1\mu^e_{pq}(i,j-1) + M_2\mu^e_{p'q'}(i,j-1))\pi(i,j-1)}{\pi(i,j)}$

5: **if** $p = i$ **then** $v^t_{pz}(i,2) = 1$
11: **if** $p = i$ **then** $M := 1$
12: $v^t_{pz}(i,j) := \frac{\sum_i (v^t_{pz}(i,j-1)\pi(i,j-1)) + (2M\mu^t_{pz}(p,j-1) + M)\pi(p,j-1)}{\pi(i,j)}$

11: **if** $p' = j$ **then** $M_1 := 1$
if $p = j$ **then** $M_2 := 1$
12: $v^t_{pzp'z'}(i,j) := \frac{(\sum_i v^t_{pzp'z'}(i,j-1)\pi(i,j-1) + M_1\mu^t_{pz}(p',j-1)\pi(p',j-1) + M_2\mu^t_{p'z'}(p,j-1)\pi(p,j-1))}{\pi(i,j)}$

11: **if** $z' = i$ **then** $M_1 := 1$
if $q = x_j \wedge p = i$ **then** $M_2 := 1$
12: $v^{et}_{pqp'z}(i,j) := \frac{(\sum_i v^{et}_{pqp'z}(i,j-1)\pi(i,j-1) + M_1\mu^e_{pq}(p',j-1)\pi(p',j-1) + M_2\mu^t_{p'z'}(p,j)\pi(p,j))}{\pi(i,j)}$

References

Y. Altun, T. Hofmann, and M. Johnson. Discriminative learning for label sequences via boosting. In *Advances in Neural Information Processing Systems (NIPS)*, pages 977-984, Vancouver, British Columbia, 2003.

Algorithm 5 Extra formulas for affine gap penalties

$$\mu_e(i,j) := \tfrac{1}{\pi(i,j)}(\mu_e(i-1,j)\pi(i-1,j)+\pi(i-2,j)+\mu_e(i,j-1)\pi(i,j-1)$$
$$+\pi(i,j-2)+\mu_e(i-1,j-1)\pi(i-1,j-1))$$

$$\mu_o(i,j) := \tfrac{1}{\pi(i,j)}(\mu_o(i-1,j)\pi(i-1,j)+\pi(i-1,j)-\pi(i-2,j)+$$
$$\mu_o(i,j-1)\pi(i,j-1)+\pi(i,j-1)-\pi(i,j-2)+\mu_o(i-1,j-1)\pi(i-1,j-1))$$

$$\nu_{oo}(i,j) := \tfrac{1}{\pi(i,j)}(\nu_{oo}(i-1,j-1)\pi(i-1,j-1)+\nu_{oo}(i-1,j)\pi(i-1,j)$$
$$+2(\mu_o(i-1,j)\pi(i-1,j)-\mu_o(i-2,j)\pi(i-2,j)-\pi(i-2,j)+\pi(i-3,j))$$
$$+\pi(i-1,j)-\pi(i-2,j)+\nu_{oo}(i,j-1)\pi(i,j-1)+2(\mu_o(i,j-1)\pi(i,j-1)$$
$$-\mu_o(i,j-2)\pi(i,j-2)-\pi(i,j-2)+\pi(i,j-3))+\pi(i,j-1)-\pi(i,j-2)$$

$$\nu_{ee}(i,j) := \tfrac{1}{\pi(i,j)}(\nu_{ee}(i-1,j-1)\pi(i-1,j-1)+\nu_{ee}(i-1,j)\pi(i-1,j)$$
$$+2\mu_e(i-2,j)\pi(i-2,j)+2\pi(i-3,j)+\pi(i-2,j)+\nu_{ee}(i,j-1)\pi(i,j-1)$$
$$+2\mu_e(i,j-2)\pi(i,j-2)+2\pi(i,j-3)+\pi(i,j-2))$$

$$\nu_{mo}(i,j) := \tfrac{1}{\pi(i,j)}((\nu_{mo}(i-1,j)+\mu_m(i-1,j))\pi(i-1,j)-\mu_m(i-2,j)\pi(i-2,j)$$
$$+(\nu_{mo}(i,j-1)+\mu_m(i,j-1))\pi(i,j-1)-\mu_m(i,j-2)\pi(i,j-2)$$
$$+(\nu_{mo}(i-1,j-1)+M\mu_o(i-1,j-1))\pi(i-1,j-1))$$

$$\nu_{me}(i,j) := \tfrac{1}{\pi(i,j)}((\nu_{me}(i-1,j-1)+M\mu_e(i-1,j-1))\pi(i-1,j-1)$$
$$+\nu_{me}(i-1,j)\pi(i-1,j)+\mu_m(i-2,j)\pi(i-2,j)+\nu_{me}(i,j-1)\pi(i,j-1)$$
$$+\mu_e(i,j-2)\pi(i,j-2))$$

$$\nu_{eo}(i,j) := \tfrac{1}{\pi(i,j)}(\nu_{eo}(i-1,j-1)\pi(i-1,j-1)+\nu_{eo}(i,j-1)\pi(i,j-1)$$
$$+\mu_o(i,j-2)\pi(i,j-2)+\pi(i,j-2)-2\pi(i,j-3)+\mu_e(i,j-1)\pi(i,j-1)$$
$$-\mu_e(i,j-2)\pi(i,j-2)+\nu_{eo}(i-1,j)\pi(i-1,j)+\mu_o(i-2,j)\pi(i-2,j)$$
$$+\pi(i-2,j)-2\pi(i-3,j)+\mu_e(i-1,j)\pi(i-1,j)-\mu_e(i-2,j)\pi(i-2,j))$$

Y. Altun, T. Hofmann, and A. J. Smola. Gaussian process classification for segmenting and annotating sequences. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, Banff, Alberta, Canada, 2004.

Y. Altun, I. Tsochantaridis, T. Hofmann. Hidden Markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 3-10, Washington, DC, USA, 2003.

P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463-482, 2002.

H.S. Booth, J.H. Maindonald, S.R. Wilson, and J.E. Gready. An efficient Z-score algorithm for assessing sequence alignments. *Journal of Computational Biology*, 11(4):616-25, 2004.

M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1-8, 2002.

M. Collins. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 489-496, 2002.

C. B. Do, D. A. Woods, and S. Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22:e90-8, 2006.

R.F. Doolittle. Similar amino acid sequences: chance or common ancestry. *Science*, 214:149-159, 1981.

R. D. Dowell and S. R. Eddy. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics*, 5(71):1-14, 2004.

S. Elizalde and K. Woods. Bounds on the number of inference functions of a graphical model. *Statistica Sinica*, 17:1395-1415, 2007.

Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, pages 170-178, Madison, Wisconson, USA, 1998.

S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S.R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31(1):439-441, 2003.

D. Gusfield, K. Balasubramanian, and D. Naor. Parametric optimization of sequence alignment. *Algorithmica*, 12:312-326, 1994.

D. Gusfield and P. Stelling. Parametric and inverse-parametric sequence alignment with XPARAL. *Methods in Enzymology*, 266:481-494, 1996.

T. Joachims, T. Galor, and R. Elber, Learning to align sequences: a maximum-margin approach, In *New Algorithms for Macromolecular Simulation*, B. Leimkuhler, LNCS Vol. 49, Springer, 2005.

J. Kececioglu and E. Kim, Simple and fast inverse alignment, In *Proceedings of the Tenth ACM Conference on Research in Computational Molecular Biology (RECOMB)*, pages 441-455, Venice, Italy, 2006.

T. Kudo. CRF++: Yet another CRF toolkit, 2005. *[http://crfpp.sourceforge.net]*.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282-289, Williamstown, MA, USA, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, pages 64-71, Banff, Alberta, Canada, 2004.

C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, 1999.

D. McAllester. Generalization bounds and consistency for structured labeling in predicting structured data. *Predicting Structured Data*, edited by G. Bakir, T. Hofmann, B. Scholkopf, A. Smola, B. Taskar, and S. V. N. Vishwanathan, MIT Press, 2007.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591-598, Stanford, CA, USA, 2000.

C. McDiarmid. On the method of bounded differences, *London Mathematical Society Lecture Note Series*, 141:148-188, 1989.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443-453, 1970.

L. Pachter and B. Sturmfels. Parametric inference for biological sequence analysis. In *Proceedings of the National Academy of Sciences USA*, 101(46):16138-16143, 2004.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257-286, 1989.

E. Ricci, T. De Bie, and N. Cristianini. Learning to align: a statistical approach. In *Proceedings of the Seventh International Symposium on Intelligent Data Analysis (IDA)*, pages 25-36, Ljubljana, Slovenia, 2007.

K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 21(Suppl 2):ii237-ii242, 2005.

R. Schapire and Y. Singer. Improved boosting algorithms using confidencerated predictions. *Machine Learning*, 37(3):297-336, 1999.

F. Sun, D. Fernandez-Baca, and W. Yu. Inverse parametric sequence alignment. *Journal of Algorithms*, 53(1):36-54, 2004.

B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In In *Advances in Neural Information Processing Systems (NIPS)*, Vancouver and Whistler, British Columbia, Canada, 2003.

B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1-8, Barcelona, Spain, 2004.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables, *Journal of Machine Learning Research*, 6(9):1453-1484, 2005.

V. N. Vapnik. *Statistical Learning Theory*. Wiley & Sons, Inc., 1998.

D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 2(10):189-208, 1967.

Journal of Machine Learning Research 9 (2008) 2847-2880 Submitted 7/08; Revised 11/08; Published 12/08

Structural Learning of Chain Graphs via Decomposition

Zongming Ma[*] ZONGMING@STANFORD.EDU
Xianchao Xie[†] XXIE@FAS.HARVARD.EDU
Zhi Geng ZGENG@MATH.PKU.EDU.CN
School of Mathematical Sciences, LMAM
Peking University
Beijing 100871, China

Editor: David Maxwell Chickering

Abstract

Chain graphs present a broad class of graphical models for description of conditional independence structures, including both Markov networks and Bayesian networks as special cases. In this paper, we propose a computationally feasible method for the structural learning of chain graphs based on the idea of decomposing the learning problem into a set of smaller scale problems on its decomposed subgraphs. The decomposition requires conditional independencies but does not require the separators to be complete subgraphs. Algorithms for both skeleton recovery and complex arrow orientation are presented. Simulations under a variety of settings demonstrate the competitive performance of our method, especially when the underlying graph is sparse.

Keywords: chain graph, conditional independence, decomposition, graphical model, structural learning

1. Introduction

Graphical models are widely used to represent and analyze conditional independencies and causal relationships among random variables. Monographs on this topic include Cowell et al. (1999), Cox and Wermuth (1996), Edwards (2000), Lauritzen (1996), Pearl (1988) and Spirtes et al. (2000). Two most well-known classes of graphical models are *Markov networks* (undirected graph) and *Bayesian networks* (directed acyclic graph). Wermuth and Lauritzen (1990) introduced the broader class of *block-recursive graphical models* (chain graph models), which includes, but is not limited to, the above two classes.

Among a multitude of research problems about graphical models, structural learning (also called model selection in statistics community) has been extensively discussed and continues to be a field of great interest. There are primarily two categories of methods: score-based methods (using AIC, BIC, posterior score, etc.) and constraint-based methods (using significance testing). Lauritzen (1996, Section 7.2) provides a good summary of the most important works done in the last century. Recent works in this area include Ravikumar et al. (2008), Friedman et al. (2007), Kalisch and Bühlmann (2007), Meinshausen and Bühlmann (2006), Tsamardinos et al. (2006), Ellis and Wong (2006), Friedman and Koller (2003), Chickering (2002), Friedman et al. (1999), etc. However, most of these studies are exclusively concerned with either Markov networks or Bayesian

∗. Also in the Department of Statistics, Stanford University, Stanford, CA 94305.
†. Also in the Department of Statistics, Harvard University, Cambridge, MA 02138.

networks. To our limited knowledge, Studený (1997) is the only work that addresses the issue of learning chain graph structures in the literature. Recently, Drton and Perlman (2008) studied the special case of Gaussian chain graph models using a multiple testing procedure, which requires prior knowledge of the dependence chain structure.

Chain graph models are most appropriate when there are both response-explanatory and symmetric association relations among variables, while Bayesian networks specifically deal with the former and Markov networks focus on the later. Given the complexity of many modern systems of interest, it is usually desirable to include both types of relations in a single model. See also Lauritzen and Richardson (2002).

As a consequence of the versatility, chain graph models have received a growing attention as a modeling tool in statistical applications recently. For instance, Stanghellini et al. (1999) constructed a chain graph model for credit scoring in a case study in finance. Carroll and Pavlovic (2006) employed chain graphs to classify proteins in bioinformatics, and Liu et al. (2005) used them to predict protein structures. However, in most applications, chain graphs are by far not as popular as Markov networks and Bayesian networks. One important reason, we believe, is the lack of readily available algorithms for chain graph structure recovery.

To learn the structure of Bayesian networks, Xie et al. (2006) proposed a 'divide-and-conquer' approach. They showed that the structural learning of the whole DAG can be decomposed into smaller scale problems on (overlapping) subgraphs. By localizing the search of d-separators, their algorithm can reduce the computational complexity greatly.

In this paper, we focus on developing a computationally feasible method for structural learning of chain graphs along with this decomposition approach. As in structural learning of a Bayesian network, our method starts with finding a decomposition of the entire variable set into subsets, on each of which the local skeleton is then recovered. However, unlike the case of Bayesian networks, the structural learning of chain graph models is more complicated due to the extended Markov property of chain graphs and the presence of both directed and undirected edges. In particular, the rule in Xie et al. (2006) for combining local structures into a global skeleton is no longer applicable and a more careful work must be done to ensure a valid combination. Moreover, the method for extending a global skeleton to a Markov equivalence class is significantly different from that for Bayesian networks.

In particular, the major contribution of the paper is twofold: (a) for learning chain graph skeletons, an algorithm is proposed which localizes the search for c-separators and has a much reduced runtime compared with the algorithm proposed in Studený (1997); (b) a polynomial runtime algorithm is given for extending the chain graph skeletons to the Markov equivalence classes. We also demonstrate the efficiency of our methods through extensive simulation studies.

The rest of the paper is organized as follows. In Section 2, we introduce necessary background for chain graph models and the concept of decomposition via separation trees. In Section 3, we present the theoretical results, followed by a detailed description of our learning algorithms. Moreover, we discuss the issue of how to construct a separation tree to represent the decomposition. The computational analysis of the algorithms are presented in Section 4. Numerical experiments are reported in Section 5 to demonstrate the performance of our method. Finally, we conclude with some discussion in Section 6. Proofs of theoretical results and correctness of algorithms are shown in Appendices.

2. Definitions and Preliminaries

In this section, we first introduce the necessary graphical model terminology in Section 2.1 and then give the formal definition of separation trees in Section 2.2.

2.1 Graphical Model Terminology

For self-containedness, we briefly introduce necessary definitions and notations in graph theory here, most of which follow those in Studený (1997) and Studený and Bouckaert (2001). For a general account, we refer the readers to Cowell et al. (1999) and Lauritzen (1996).

A graph $\mathcal{G} = (V, E)$ consists of a vertex set V and an edge set E. For vertices $u, v \in V$, we say that there is an undirected edge $u - v$ if $(u, v) \in E$ and $(v, u) \in E$. If $(u, v) \in E$ and $(v, u) \notin E$, we say that there is a directed edge from u to v and write $u \to v$. We call undirected edges *lines* and directed edges *arrows*. The *skeleton* $\mathcal{G}'$ of a graph $\mathcal{G}$ is the undirected graph obtained by replacing the arrows of $\mathcal{G}$ by lines. If every edge in graph $\mathcal{G}$ is undirected, then for any vertex u, we define the *neighborhood* $\mathrm{ne}_{\mathcal{G}}(u)$ of u to be the set of vertices v such that $u - v$ in $\mathcal{G}$.

A *route* in $\mathcal{G}$ is a sequence of vertices $(v_0, \cdots, v_k), k \geq 0$, such that $(v_{i-1}, v_i) \in E$ or $(v_i, v_{i-1}) \in E$ for $i = 1, \cdots, k$, and the vertices v_0 and v_k are called *terminals* of the route. It is called *descending* if $(v_{i-1}, v_i) \in E$ for $i = 1, \cdots, k$. We write $u \mapsto v$ if there exists a descending route from u to v. If $v_0, \cdots, v_k$ are distinct vertices, the route is called a *path*. A route is called a *pseudocycle* if $v_0 = v_k$, and a *cycle* if further $k \geq 3$ and $v_0, \cdots, v_{k-1}$ are distinct. A (pseudo) cycle is *directed* if it is descending and there exists at least one $i \in \{1, \cdots, k\}$, such that $(v_i, v_{i-1}) \notin E$.

A graph with only undirected edges is called an *undirected graph* (UG). A graph with only directed edges and without directed cycles is called a *directed acyclic graph* (DAG). A graph that has no directed (pseudo) cycles is called a *chain graph*.

By a *section* of a route $\rho = (v_0, \cdots, v_k)$ in $\mathcal{G}$, we mean a maximal undirected subroute $\sigma : v_i - \cdots - v_j,\ 0 \leq i \leq j \leq k$ of ρ. The vertices v_i and v_j are called the *terminals* of the section σ. Further the vertex v_i (or v_j) is called a *head-terminal* if $i > 0$ and $v_{i-1} \to v_i$ in $\mathcal{G}$ (or $j < k$ and $v_j \leftarrow v_{j+1}$ in $\mathcal{G}$), otherwise it is called a *tail-terminal*. A section σ of a route ρ is called a *head-to-head* section with respect to ρ if it has two head-terminals, otherwise it is called *non head-to-head*. For a set of vertices $S \subset V$, we say that a section $\sigma : v_i - \cdots - v_j$ is *outside* S if $\{v_i, \cdots, v_j\} \cap S = \emptyset$. Otherwise we say that σ is *hit* by S.

A *complex* in $\mathcal{G}$ is a path $(v_0, \cdots, v_k),\ k \geq 2$, such that $v_0 \to v_1$, $v_i - v_{i+1}$ (for $i = 1, \cdots, k-2$) and $v_{k-1} \leftarrow v_k$ in $\mathcal{G}$, and no additional edge exists among vertices $\{v_0, \cdots, v_k\}$ in $\mathcal{G}$. We call the vertices v_0 and v_k the *parents* of the complex and the set $\{v_1, \cdots, v_{k-1}\}$ the *region* of the complex. The set of parents of a complex κ is denoted by $\mathrm{par}(\kappa)$. Note that the concept of complex was proposed in Studený (1997) and is equivalent to the notion of 'minimal complex' in Frydenberg (1990).

An arrow in a graph $\mathcal{G}$ is called a *complex arrow* if it belongs to a complex in $\mathcal{G}$. The *pattern* of $\mathcal{G}$, denoted by $\mathcal{G}^*$, is the graph obtained by turning the arrows that are not in any complex of $\mathcal{G}$ into lines. The *moral graph* $\mathcal{G}^m$ of $\mathcal{G}$ is the graph obtained by first joining the parents of each complex by a line and then turning arrows of the obtained graph into lines.

Studený and Bouckaert (2001) introduced the notion of *c-separation* for chain graph models. Hereunder we introduce the concept in the form that facilitates the proofs of our results. We say that a route ρ on $\mathcal{G}$ is *intervented* by a subset S of V if and only if there exists a section σ of ρ such that:

1. either σ is a head-to-head section with respect to ρ, and σ is outside S; or

2. σ is a non head-to-head section with respect to ρ, and σ is hit by S.

Definition 1 *Let A, B, S be three disjoint subsets of the vertex set V of a chain graph $\mathcal{G}$, such that A, B are nonempty. We say that A and B are* c-separated by S on $\mathcal{G}$, *written as $\langle A, B \,|\, S \rangle_{\mathcal{G}}^{sep}$, if every route with one of its terminals in A and the other in B is intervented by S. We also call S a* c-separator *for A and B.*

For a chain graph $\mathcal{G} = (V, E)$, let each $v \in V$ be associated with a random variable X_v with domain $\mathcal{X}_v$ and μ the underlying probability measure on $\prod_{v \in V} \mathcal{X}_v$. A probability distribution P on $\prod_{v \in V} \mathcal{X}_v$ is *strictly positive* if $\mathrm{d}P(x)/\mathrm{d}\mu > 0$ for any $x \in \prod_{v \in V} \mathcal{X}_v$. From now on, all probability distributions considered are assumed to be strictly positive. A probability distribution P on $\prod_{v \in V} \mathcal{X}_v$ is *faithful* with respect to $\mathcal{G}$ if for any triple (A, B, S) of disjoint subsets of V such that A and B are non-empty, we have

$$\langle A, B \,|\, S \rangle_{\mathcal{G}}^{sep} \;\Leftrightarrow\; X_A \perp\!\!\!\perp X_B \,|\, X_S \,, \tag{1}$$

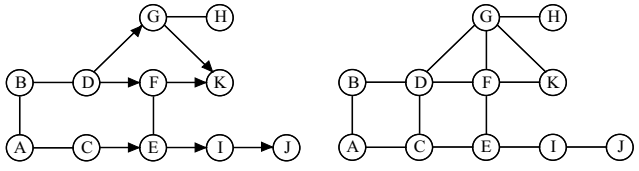
where $X_A = \{X_v : v \in A\}$ and $X_A \perp\!\!\!\perp X_B | X_S$ means the conditional independency of X_A and X_B given X_S; P is *Markovian* with respect to $\mathcal{G}$ if (1) is weakened to

$$\langle A, B \,|\, S \rangle_{\mathcal{G}}^{sep} \;\Rightarrow\; X_A \perp\!\!\!\perp X_B \,|\, X_S \,.$$

In the rest of the paper, $A \perp\!\!\!\perp B | S$ is used as short notation for $X_A \perp\!\!\!\perp X_B | X_S$ when confusion is unlikely.

It is known that chain graphs can be classified into *Markov equivalence classes*, and those in the same equivalence class share the same set of Markovian distributions. The following result from Frydenberg (1990) characterizes equivalence classes graphically: Two chain graphs are Markov equivalent if and only if they have the same skeleton and complexes, that is, they have the same pattern. The following example illustrates some of the concepts that are introduced above.

Example 1. Consider the chain graph $\mathcal{G}$ in Fig. 1(a). $D \to F - E \leftarrow C$ and $F \to K \leftarrow G$ are the two complexes. The route $\rho = (D, F, E, I, E, C)$ is intervented by an empty set since the head-to-head section $(E \to) I (\leftarrow E)$ is outside the empty set. It is also intervented by E since the non head-to-head section $(D \to) F - E (\to I)$ is hit by E. However, D and C are not c-separated by E since the route (D, F, E, C) is not intervented by E. The moral graph $\mathcal{G}^m$ of $\mathcal{G}$ is shown in Fig. 1(b), where edges $C - D$ and $F - G$ are added due to moralization. ■



(a) (b)

Figure 1: (a) a chain graph $\mathcal{G}$; (b) its moral graph $\mathcal{G}^m$.

2.2 Separation Trees

In this subsection, we introduce the notion of *separation trees* which is used to facilitate the representation of the decomposition. The concept is similar to the junction tree of cliques and the independence tree introduced for DAG as 'd-separation trees'(Xie et al., 2006).

Let $\mathcal{C} = \{C_1, \cdots, C_H\}$ be a collection of distinct variable sets such that for $h = 1, \cdots, H$, $C_h \subseteq V$. Let $\mathcal{T}$ be a tree where each node corresponds to a distinct variable set in $\mathcal{C}$, to be displayed as a triangle (see, for example, Fig. 2). The term 'node' is used for a separation tree to distinguish from the term 'vertex' for a graph in general. An undirected edge $e = (C_i, C_j)$ connecting nodes C_i and C_j in $\mathcal{T}$ is attached with a *separator* $S = C_i \cap C_j$, which is displayed as a rectangle. A separator S is *connected to a node* C if there is some other node C', such that S attaches to the edge (C, C'). Removing an edge e or equivalently, removing a separator S from $\mathcal{T}$ splits $\mathcal{T}$ into two subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ with node sets $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. We use $V_i = \cup_{C \in \mathcal{C}_i} C$ to denote the unions of the vertices contained in the nodes of the subtree $\mathcal{T}_i$ for $i = 1, 2$.

Definition 2 *A tree $\mathcal{T}$ with node set $\mathcal{C}$ is said to be a* separation tree *for a chain graph $\mathcal{G} = (V, E)$ if*

1. $\cup_{C \in \mathcal{C}} C = V$, *and*
2. *for any separator S in $\mathcal{T}$ with V_1 and V_2 defined as above by removing S, we have $\langle V_1 \backslash S, V_2 \backslash S \,|\, S \rangle_{\mathcal{G}}^{sep}$.*

Notice that a separator is defined in terms of a tree whose nodes consist of variable sets, while the c-separator is defined based on a chain graph. In general, these two concepts are not related, though for a separation tree its separator must be some corresponding c-separator in the underlying chain graph.

Example 1. (Continued). Suppose that

$$\mathcal{C} = \{\{A,B,C\}, \{B,C,D\}, \{C,D,E\}, \{D,E,F\}, \{E,I,J\}, \{D,F,G,H,K\}\}$$

is a collection of vertex sets. A separation tree $\mathcal{T}$ of $\mathcal{G}$ in Fig. 1(a) with node set $\mathcal{C}$ is shown in Fig. 2. If we delete the separator $\{D,E\}$, we obtain two subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ with node sets $\mathcal{C}_1 = \{\{A,B,C\}, \{B,C,D\}, \{C,D,E\}\}$ and $\mathcal{C}_2 = \{\{D,E,F\}, \{E,I,J\}, \{D,F,G,H,K\}\}$. In $\mathcal{G}$, the separator $S = \{D,E\}$ c-separates $V_1 \backslash S = \{A,B,C\}$ and $V_2 \backslash S = \{F,G,H,I,J,K\}$. ■

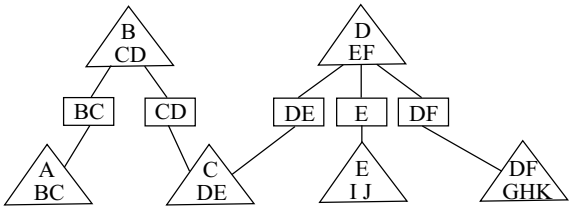


Figure 2: A separation $\mathcal{T}$ of the graph $\mathcal{G}$ in Fig. 1(a).

Not surprisingly, the separation tree could be regarded as a scheme for decomposing the knowledge represented by the chain graph into local subsets. Reciprocally, given a separation tree, we

can combine the information obtained locally to recover the global information, an idea that will be formalized and discussed in subsequent sections.

The definition of separation trees for chain graphs is similar to that of junction trees of cliques, see Cowell et al. (1999) and Lauritzen (1996). Actually, it is not difficult to see that a junction tree of a chain graph $\mathcal{G}$ is also its separation tree. However, we point out two differences here: (a) a separation tree is defined with *c*-separation and it does not require that every node is a clique or that every separator is complete on the moral graph; (b) junction trees are mostly used as inference engines, while our interest in separation trees is mainly derived from its power in facilitating the decomposition of structural learning.

3. Structural Learning of Chain Graphs

In this section, we discuss how separation trees can be used to facilitate the decomposition of the structural learning of chain graphs. Theoretical results are presented first, followed by descriptions of several algorithms that are the summary of the key results in our paper. It should be emphasized that even with perfect knowledge on the underlying distribution, any two chain graph structures in the same equivalence class are indistinguishable. Thus, we can only expect to recover a pattern from the observed data, that is, an equivalence class to which the underlying chain graph belongs. To this end, we provide two algorithms: one addresses the issue of learning the skeleton and the other focuses on extending the learned skeleton to an equivalence class. We also discuss at the end of the section the problem of constructing separation trees. Throughout the rest of the paper, we assume that any distribution under consideration is faithful with respect to some underlying chain graph.

3.1 Theoretical Results

It is known that for P faithful to a chain graph $\mathcal{G} = (V,E)$, an edge (u,v) is in the skeleton $\mathcal{G}'$ if and only if $X_u \not\perp\!\!\!\perp X_v \,|\, X_S$ for any $S \subseteq V \setminus \{u,v\}$ (see Studený 1997, Lemma 3.2 for a proof). Therefore, learning the skeleton of $\mathcal{G}$ reduces to searching for *c*-separators for all vertex pairs. The following theorem shows that with a separation tree, one can localize the search into one node or a small number of tree nodes.

Theorem 3 *Let $\mathcal{T}$ be a separation tree for a chain graph $\mathcal{G}$. Then vertices u and v are c-separated by some set $S_{uv} \subset V$ in $\mathcal{G}$ if and only if one of the following conditions hold:*

1. *u and v are not contained together in any node C of $\mathcal{T}$,*
2. *u and v are contained together in some node C, but for any separator S connected to C, $\{u,v\} \nsubseteq S$, and there exists $S'_{uv} \subseteq C$ such that $\langle u, v \,|\, S'_{uv} \rangle^{sep}_{\mathcal{G}}$,*
3. *u and v are contained together in a node C and both of them belong to some separator connected to C, but there is a subset S'_{uv} of either $\bigcup_{u \in C'} C'$ or $\bigcup_{v \in C'} C'$ such that $\langle u, v \,|\, S'_{uv} \rangle^{sep}_{\mathcal{G}}$.*

For DAGs, condition 3 in Theorem 3 is unnecessary, see Xie et al. (2006, Theorem 1). However, the example below indicates that this is no longer the case for chain graphs.

Example 1. (Continued). Consider the chain graph in Fig. 1(a) and its separation tree in Fig. 2. Let $u = D$ and $v = E$. The tree nodes containing u and v together are $\{C,D,E\}$ and $\{D,E,F\}$. Since the

separator $\{D,E\}$ is connected to both of them, neither condition 1 nor 2 in Theorem 3 is satisfied. Moreover, $u \not\perp\!\!\!\perp v|S$ for $S = \emptyset, \{C\}$ or $\{F\}$. However, we have $u \perp\!\!\!\perp v|\{C,F\}$. Since $\{C,F\} = S'$ is a subset of both $\bigcup_{u\in C'} C' = \{B,C,D\} \cup \{C,D,E\} \cup \{D,E,F\} \cup \{D,F,G,H,K\} = \{B,C,D,E,F,G,H,K\}$ and $\bigcup_{v\in C'} C' = \{C,D,E\} \cup \{D,E,F\} \cup \{E,I,J\} = \{C,D,E,F,I,J\}$, condition 3 of Theorem 3 applies. ■

Given the skeleton of a chain graph, we extend it into the equivalence class by identifying and directing all the complex arrows, to which the following result is helpful.

Proposition 4 *Let $\mathcal{G}$ be a chain graph and $\mathcal{T}$ a separation tree of $\mathcal{G}$. For any complex κ in $\mathcal{G}$, there exists some tree node C of $\mathcal{T}$ such that* $\mathrm{par}(\kappa) \subseteq C$.

By Proposition 4, to identify the parents for each complex, we need only pay attention to those vertex pairs contained in each tree node.

3.2 Skeleton Recovery with Separation Tree

In this subsection, we propose an algorithm based on Theorem 3 for the identification of chain graph skeleton with separation tree information.

Let $\mathcal{G}$ be an unknown chain graph of interest and P be a probability distribution that is faithful to it. The algorithm, summarized as Algorithm 1, is written in its population version, where we assume perfect knowledge of all the conditional independencies induced by P.

Algorithm 1 consists of three main parts. In part 1 (lines 2-10), local skeletons are recovered in each individual node of the input separation tree. By condition 1 of Theorem 3, edges deleted in any local skeleton are also absent in the global skeleton. In part 2 (lines 11-16), we combine all the information obtained locally into a partially recovered global skeleton which may have extra edges not belonging to the true skeleton. Finally, we eliminate such extra edges in part 3 (lines 17-22).

The correctness of Algorithm 1 is proved in Appendix B. We conclude this subsection with some remarks on the algorithm.

Remarks on Algorithm 1:

1. Although we assume perfect conditional independence knowledge in Algorithm 1, it remains valid when these conditional independence relations are obtained via performing hypotheses tests on data generated over P as in most real-world problems. When this is the case, we encounter the problem of multiple testing. See Sections 3.4 and 5 for more details.

2. The c-separator set $\mathcal{S}$ is not necessary for skeleton recovery. However, it will be useful later in Section 3.3 when we try to recover the pattern based on the learned skeleton.

3. Part 3 of Algorithm 1 is indispensable as is illustrated by the following example.

Example 1. (Continued) We apply Algorithm 1 to the chain graph $\mathcal{G}$ in Fig. 1(a) and the corresponding separation tree given in Fig. 2. The local skeletons recovered in part 1 of the algorithm are shown in Fig. 3(a). The c-separators found in this part are $S_{BC} = \{A\}$, $S_{CD} = \{B\}$, $S_{EJ} = \{I\}$, $S_{DK} = \{F,G\}$, $S_{DH} = \{F,G\}$, $S_{FG} = \{D\}$, $S_{FH} = \{G\}$ and $S_{HK} = \{G\}$. In part 2, the local skeletons are combined by deleting the edges that are absent in at least one of the local skeletons, leading to the result in Fig. 3(b). The edge $B - C$ is deleted since it is absent in the local skeleton for the tree

Algorithm 1: Skeleton Recovery with a Separation Tree.

Input: A separation tree $\mathcal{T}$ of $\mathcal{G}$; perfect conditional independence knowledge about P.
Output: The skeleton $\mathcal{G}'$ of $\mathcal{G}$; a set $\mathcal{S}$ of c-separators.

1 Set $\mathcal{S} = \emptyset$;
2 **foreach** *Tree node* C_h **do**
3 Start from a complete undirected graph $\mathcal{G}_h$ with vertex set C_h;
4 **foreach** *Vertex pair* $\{u,v\} \subset C_h$ **do**
5 **if** $\exists S_{uv} \subset C_h$ *s.t.* $u \perp\!\!\!\perp v | S_{uv}$ **then**
6 Delete the edge (u,v) in $\mathcal{G}_h$;
7 Add S_{uv} to $\mathcal{S}$;
8 **end**
9 **end**
10 **end**
11 Combine the graphs $\mathcal{G}_h = (C_h, E_h), h = 1, \cdots, H$ into an undirected graph $\mathcal{G}' = (V, \cup_{h=1}^{H} E_h)$;
12 **foreach** *Vertex pair* $\{u,v\}$ *contained in more than one tree node and* $(u,v) \in \mathcal{G}'$ **do**
13 **if** $\exists C_h$ *s.t.* $\{u,v\} \subset C_h$ *and* $(u,v) \notin E_h$ **then**
14 Delete the edge (u,v) in $\mathcal{G}'$;
15 **end**
16 **end**
17 **foreach** *Vertex pair* $\{u,v\}$ *contained in more than one tree node and* $(u,v) \in \mathcal{G}'$ **do**
18 **if** $u \perp\!\!\!\perp v | S_{uv}$ *for some* $S_{uv} \subset \mathrm{ne}_{\mathcal{G}'}(u)$ *or* $\mathrm{ne}_{\mathcal{G}'}(v)$ *such that it is not a subset of any* C_h *with* $\{u,v\} \subset C_h$ **then**
19 Delete the edge (u,v) in $\mathcal{G}'$;
20 Add S_{uv} to $\mathcal{S}$;
21 **end**
22 **end**

node $\{A,B,C\}$. In part 3, we need to check $D-E$ and $D-F$. $D-E$ is deleted since $D \perp\!\!\!\perp E | \{C,F\}$, and we record $S_{DE} = \{C,F\}$. The recovered skeleton is finally shown in Fig. 3(c). ■

3.3 Complex Recovery

In this subsection, we propose Algorithm 2 for finding and orienting the complex arrows of $\mathcal{G}$ after obtaining the skeleton $\mathcal{G}'$ in Algorithm 1. We call this stage of structural learning *complex recovery*. As in the previous subsection, we assume separation tree information and perfect conditional independence knowledge. For simplicity, let S^c denote $V \backslash S$ for any vertex set $S \subseteq V$.

Example 1. (Continued) After performing Algorithm 1 to recover the skeleton of $\mathcal{G}$, we apply Algorithm 2 to find and orient all the complex arrows. For example, when we pick $[C,D]$ in line 2 of Algorithm 2 and consider $C-E$ in line 3 for the inner loop, we find $S_{CD} = \{B\}$ and $C \not\perp\!\!\!\perp D | S_{CD} \cup \{E\}$. Therefore, we orient $C \to E$ in line 5. Similarly, we orient $D \to F, F \to K$ and $G \to K$ when considering $[D,C]$ with $D-F$ (in the inner loop), $[F,G]$ with $F-K$ and $[G,F]$ with $G-K$ and observing that $C \not\perp\!\!\!\perp D | S_{CD} \cup \{F\}$ and $F \not\perp\!\!\!\perp G | S_{FG} \cup \{K\}$, where the c-separators $S_{CD} = \{B\}$ and

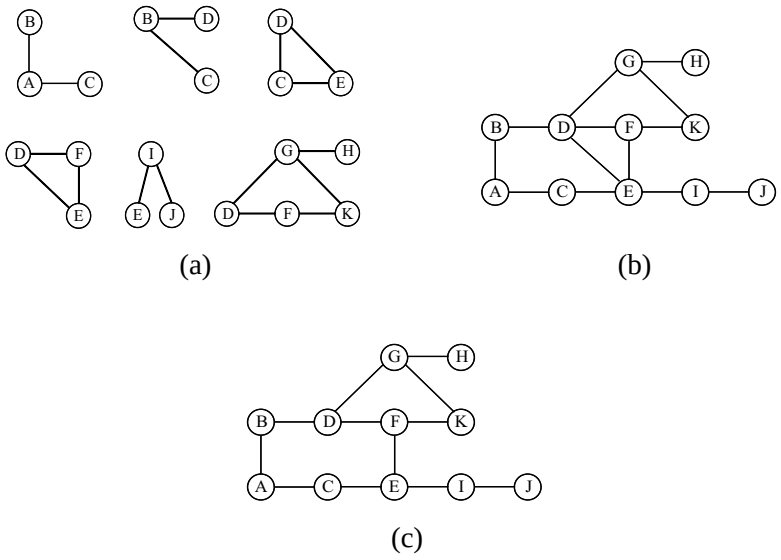


Figure 3: (a) local skeletons recovered in part 1 of Algorithm 1 for all nodes of $\mathcal{T}$ in Fig. 2; (b) partially recovered global skeleton of $\mathcal{G}$ in part 2 of Algorithm 1; (c) completely recovered global skeleton of $\mathcal{G}$ in part 3 of Algorithm 1.

Algorithm 2: Complex Recovery.

Input: Perfect conditional independence knowledge; the skeleton $\mathcal{G}'$ and the set $\mathcal{S}$ of c-separators obtained in Algorithm 1.

Output: The pattern $\mathcal{G}^*$ of $\mathcal{G}$.

```
Initialize G* = G';
foreach Ordered pair [u, v] such that S_uv ∈ S do
    foreach u − w in G* do
        if u ⊥̸ v | S_uv ∪ {w} then
            Orient u − w as u → w in G*;
        end
    end
end
Take the pattern of G*.
```

$S_{FG} = \{K\}$ were obtained during the execution of Algorithm 1. We do not orient any other edge in Algorithm 2. The resulting graph is shown in Fig. 4, which is exactly the pattern for our original chain graph $\mathcal{G}$ in Fig. 1(a). ■

The correctness of Algorithm 2 is guaranteed by Proposition 4. For the proof, see Appendix B.

Remarks on Algorithm 2:

1. As Algorithm 1, Algorithm 2 is valid when independence test is correctly performed using data and again, multiple testing becomes an important issue here.

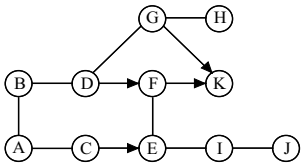


Figure 4: The pattern of $\mathcal{G}$ recovered by applying Algorithm 2.

2. In Algorithm 2, the set $\mathcal{S}$ obtained in Algorithm 1 helps avoid the computationally expensive looping procedure taken in the pattern recovery algorithm in Studený (1997) for complex arrow orientation.

3. Line 9 of Algorithm 2 is necessary for chain graphs in general, see Example 2 below.

4. To get the pattern of $\mathcal{G}^*$ in line 9, at each step, we consider a pair of candidate complex arrows $u_1 \to w_1$ and $u_2 \to w_2$ with $u_1 \neq u_2$, then we check whether there is an undirected path from w_1 to w_2 such that none of its intermediate vertices is adjacent to either u_1 or u_2. If there exists such a path, then $u_1 \to w_1$ and $u_2 \to w_2$ are labeled (as complex arrows). We repeat this procedure until all possible candidate pairs are examined. The pattern is then obtained by removing directions of all unlabeled arrows in $\mathcal{G}^*$.

Example 2. Consider the chain graph $\tilde{\mathcal{G}}$ in Fig. 5(a) and the corresponding separation tree $\tilde{\mathcal{T}}$ in Fig. 5(e). After applying Algorithm 1, we obtain the skeleton $\tilde{\mathcal{G}}'$ of $\tilde{\mathcal{G}}$. In the execution of Algorithm 2, when we pick $[B,F]$ in line 2 and A in line 3, we have $B \perp\!\!\!\perp F | \emptyset$, that is, $S_{BF} = \emptyset$, and find that $B \not\perp\!\!\!\perp F | A$. Hence we orient $B - A$ as $B \to A$ in line 5, which is not a complex arrow in $\tilde{\mathcal{G}}$. Note that we do not orient $A - B$ as $A \to B$: the only chance we might do so is when $u = A$, $v = F$ and $w = B$ in the inner loop of Algorithm 2, but we have $B \in S_{AF}$ and the condition in line 4 is not satisfied. Hence, the graph we obtain before the last step in Algorithm 2 must be the one given in Fig. 5(c), which differs from the recovered pattern in Fig. 5(d). This illustrates the necessity of the last step in Algorithm 2. To see how the edge $B \to A$ is removed in the last step of Algorithm 2, we observe that, if we follow the procedure described in Remark 4 on Algorithm 2, the only chance that $B \to A$ becomes one of the candidate complex arrow pair is when it is considered together with $F \to D$. However, the only undirected path between A and D is simply $A - D$ with D adjacent to B. Hence $B \to A$ stays unlabeled and will finally get removed in the last step of Algorithm 2. ■

3.4 Sample Versions of Algorithms 1 and 2

In this section, we present a brief description on how to obtain a sample version of the previous algorithms and the related issues.

To apply the previous methods to a data set, we proceed in the exactly same way as before. The only difference in the sample version of the algorithms lies in that statistical hypothesis tests are needed to evaluate the predicate on line 5, 13 and 19 in Algorithm 1 and on line 4 in Algorithm 2. Specifically, conditional independence test of two variables u and v given a set C of variables is required. Let the null hypothesis H_0 be $u \perp\!\!\!\perp v | C$ and alternative H_1 be that H_0 may not hold. Generally

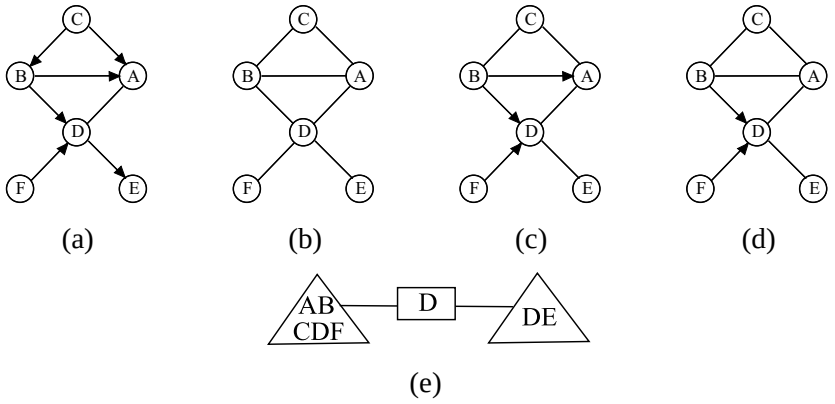


Figure 5: (a) the chain graph $\tilde{\mathcal{G}}$ in Example 2; (b) the skeleton $\tilde{\mathcal{G}}'$ of $\tilde{\mathcal{G}}$; (c) the graphical structure $\tilde{\mathcal{G}}^*$ before executing the last line in Algorithm 2; (d) the graphical structure $\tilde{\mathcal{G}}^*$ obtained after executing Algorithm 2; (e) a separation tree $\tilde{\mathcal{T}}$ for $\tilde{\mathcal{G}}$ in (a).

we can use the likelihood ratio test statistic

$$G^2 = -2\log\frac{\sup\{L(\theta|D) \text{ under } H_0\}}{\sup\{L(\theta|D) \text{ under } H_1\}},$$

where $L(\theta|D)$ is the likelihood function of parameter θ with observed data D. Under H_0, the statistic G^2 asymptotically follows the χ^2 distribution with df degrees of freedom being equal to the difference of the dimensions of parameters for the alternative and null hypothesis (Wilks, 1938).

Let $\mathbf{X}_k$ be a vector of variables and N be the sample size. For the case of a Gaussian distribution, the test statistic for testing $X_i \perp\!\!\!\perp X_j|\mathbf{X}_k$ can be simplified to

$$\begin{aligned} G^2 &= -N \times \log(1-\text{corr}^2(X_i, X_j|\mathbf{X}_k)) \\ &= N \times \log\frac{\det(\hat{\Sigma}_{\{i,k\}\{i,k\}})\det(\hat{\Sigma}_{\{j,k\}\{j,k\}})}{\det(\hat{\Sigma}_{\{i,j,k\}\{i,j,k\}})\det(\hat{\Sigma}_{k,k})}, \end{aligned}$$

which has an asymptotic χ^2 distribution with $df = 1$. Actually, the exact null distribution or a better approximate distribution of G^2 can be obtained based on Bartlett decomposition, see Whittaker (1990) for details.

For the discrete case, let N_s^m be the observed frequency in a cell of $X_s = m$ where s is an index set of variables and m is category of variables X_s. For example, N_{ijk}^{abc} denotes the frequency of $X_i = a$, $X_j = b$ and $\mathbf{X}_k = c$. The G^2 statistic for testing $X_i \perp\!\!\!\perp X_j|\mathbf{X}_k$ is then given by

$$G^2 = 2\sum_{a,b,c} N_{ijk}^{abc} \log\frac{N_{ijk}^{abc} N_k^c}{N_{ik}^{ac} N_{jk}^{bc}},$$

which is asymptotically distributed as a χ^2 distribution under H_0 with degree of freedom

$$\text{df} = (\#(X_i)-1)(\#(X_j)-1)\prod_{X_l \in \mathbf{X}_k} \#(X_l),$$

where $\#(X)$ is the number of categories of variable X.

3.4.1 THE MULTIPLE TESTING PROBLEM

As mentioned in the remarks of Algorithms 1 and 2, when perfect knowledge of the population conditional independence structure is not available, we turn to hypotheses testing for obtaining these conditional independence relations from data and run into the problem of multiple testing. This is because we need to test the existence of separators for multiple edges, and we typically consider more than one candidate separators for each edge.

Although a theoretical analysis of the impact of the multiple testing problem on the overall error rate (see, for example, Drton and Perlman 2008) for the proposed method is beyond the scope of this paper, simulation studies are conducted on randomly generated chain graphs to show the impact of choices of different significance levels on our method in Section 5.1. Based on our empirical study there, we feel that choosing a small significance level (e.g., $\alpha = 0.005$ or 0.01) for the individual tests usually yields good and stable results when the sample size is reasonably large and the underlying graph is sparse.

3.5 Construction of Separation Trees

Algorithms 1 and 2 depend on the information encoded in the separation tree. In the subsection, we address the issue of how to construct a separation tree.

As proposed by Xie et al. (2006), one can construct a d-separation tree from observed data, from domain or prior knowledge of conditional independence relations or from a collection of databases. Their arguments are also valid in the current setting. In the rest of this subsection, we first extend Theorem 2 of Xie et al. (2006), which guarantees that their method for constructing a separation tree from data is valid for chain graph models. Then we propose an algorithm for constructing a separation tree from background knowledge encoded in a labeled block ordering (Roverato and La Rocca, 2006) of the underlying chain graph. We remark that Algorithm 2 of Xie et al. (2006), which constructs d-separation trees from hyper-graphs, also works in the current context.

3.5.1 FROM UNDIRECTED INDEPENDENCE GRAPH TO SEPARATION TREE

For a chain graph $\mathcal{G} = (V,E)$ and a faithful distribution P, an undirected graph $\mathcal{U}$ with a vertex set V is an *undirected independence graph* (UIG) of $\mathcal{G}$ if for any $u,v \in V$,

$$(u,v) \text{ is not an edge of } \mathcal{U} \;\Rightarrow\; u \perp\!\!\!\perp v \mid V \setminus \{u,v\} \text{ in } P. \tag{2}$$

We generalize Theorem 2 of Xie et al. (2006) as follows.

Theorem 5 *Let $\mathcal{G}$ be a chain graph. Then a junction tree constructed from any undirected independence graph of $\mathcal{G}$ is a separation tree for $\mathcal{G}$.*

Since there are standard algorithms for constructing junction trees from UIGs (see Cowell et al. 1999, Chapter 4, Section 4), the construction of separation trees reduces to the construction of UIGs. In this sense, Theorem 5 enables us to exploit various techniques for learning UIGs to serve our purpose.

As suggested by relation (2), one way of learning UIGs from data is testing the required conditional independencies. Agresti (2002) and Anderson (2003) provides general techniques for discrete and Gaussian data respectively. Recently, there are also works on estimating UIGs via ℓ_1-regularization, see, for instance, Friedman et al. (2007) for Gaussian data and Ravikumar et al.

(2008) for the discrete case. Edwards (2000, Chapter 6) presents some other established methods for UIG learning, including those that are valid when X_V includes both continuous and discrete variables. All these methods can be used to construct separation trees from data.

3.5.2 FROM LABELED BLOCK ORDERING TO SEPARATION TREE

When learning graphical models, it is a common practice to incorporate substantive background knowledge in structural learning, mostly to reduce the dimensionality of the search spaces for both computational and subject matter reasons.

Recently, Roverato and La Rocca (2006) studied in detail a general type of background knowledge for chain graphs, which they called *labeled block ordering*. We introduce this concept here and investigate how it is related to the construction of separation trees.

By summarizing Definitions 5 and 6 in Roverato and La Rocca (2006), we define the labeled block ordering as the following.

Definition 6 *Let $V_1, \cdots, V_k$ be a partition of a set of vertices V. A* labeled block ordering $\mathcal{B}$ *of V is a sequence $(V_i^{l_i}, i = 1, \cdots, k)$ where $l_i \in \{u, d, g\}$ and with the convention that $V_i = V_i^g$. We say a chain graph $\mathcal{G} = (V, E)$ is* $\mathcal{B}$-consistent *if*

1. *every edge connecting vertices $A \in V_i$ and $B \in V_j$ for $i \neq j$ is oriented from $A \to B$ if $i < j$;*
2. *for every $l_i = u$, the subgraph $\mathcal{G}_{V_i}$ is a UG;*
3. *for every $l_i = d$, the subgraph $\mathcal{G}_{V_i}$ is a DAG;*
4. *for every $l_i = g$, the subgraph $\mathcal{G}_{V_i}$ may have both directed and undirected edges.*

Example 1. (Continued) Let $V_1 = \{A, B, C, D, E, F\}$, $V_2 = \{G, H, K\}$ and $V_3 = \{I, J\}$. Then Fig. 6 shows that for a labeled block ordering $\mathcal{B} = (V_1^g, V_2^g, V_3^d)$, $\mathcal{G}$ in Fig. 1(a) is $\mathcal{B}$-consistent. ■

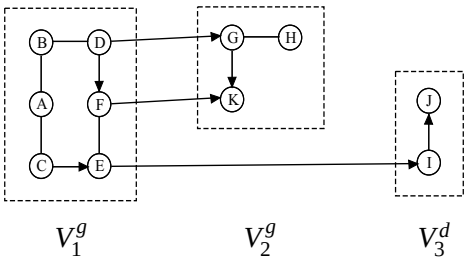


Figure 6: A labeled block ordering $\mathcal{B} = (V_1^g, V_2^g, V_3^d)$ for which $\mathcal{G}$ in Fig. 1(a) is $\mathcal{B}$-consistent.

To show how labeled block ordering helps learning separation trees, we start with the following simple example.

Example 3. Suppose that the underlying chain graph is the one in Fig. 7(a) which is $\mathcal{B}$-consistent with $\mathcal{B} = \{V_1^g, V_2^g, V_3^g\}$. Then for $V_1 = \{A, B\}, V_2 = \{C, D\}$ and $V_3 = \{E, F\}$, we have $V_1 \perp\!\!\!\perp V_3 | V_2$. Together with the total ordering on them, we can construct the DAG in Fig. 7(b) with vertices as

blocks $V_i,\ i = 1,2,3$, which depicts the conditional independence structures on the three blocks. By taking the junction tree of this DAG and replacing the blocks by the vertices they contain, we obtain the tree structure in Fig. 7(c). This is a separation tree of the chain graph in Fig. 7(a). ■

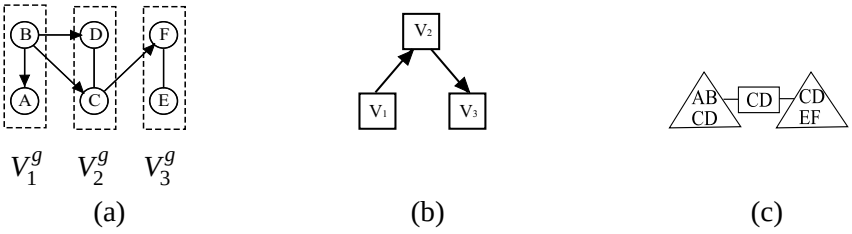


Figure 7: (a) a chain graph with a labeled block ordering; (b) the DAG constructed for the blocks; (c) the tree structure obtained from the junction tree of the DAG in (b).

Below we propose Algorithm 3 for constructing a separation tree from labeled block ordering knowledge. The idea is motivated by the preceding toy example. By omitting independence structures within each block, we can treat each block as a vertex in a DAG. The total ordering on the blocks and the conditional independence structures among them enable us to build a DAG on these blocks. By taking the junction tree of this particular DAG, we obtain a separation tree of the underlying chain graph model.

Algorithm 3: Separation Tree Construction with Labeled Block Ordering.

Input: A labeled block ordering $\mathcal{B} = (V_i^{l_i}, i = 1, \cdots, k)$ of V; perfect conditional independence knowledge.

Output: A separation tree $\mathcal{T}$ of $\mathcal{G}$.

1 Construct a DAG $\mathcal{D}$ with blocks $V_i, i = 1, \cdots, k$;
2 Construct a junction tree $\mathcal{T}$ by triangulating $\mathcal{D}$;
3 In $\mathcal{T}$, replace each block V_i by the original vertices it contains.

We note that a labeled block ordering gives a total ordering of the blocks $V_i, i = 1, \cdots, k$. Given a total ordering of the vertices of a DAG, one can use the Wermuth-Lauritzen algorithm (Spirtes et al., 2000; Wermuth and Lauritzen, 1983) for constructing (the equivalence class of) the DAG from conditional independence relations. Spirtes et al. (2000) also discussed how to incorporate the total ordering information in the PC algorithm for constructing DAGs from conditional independence relations. Since these approaches rely only on conditional independence relations, they can be used in Algorithm 3 for constructing the DAG $\mathcal{D}$. The only difference is that each vertex represents a random vector rather than a random variable now. The correctness of Algorithm 3 is proved in Appendix B.

3.5.3 SEPARATION TREE REFINEMENT

In practice, the nodes of a separation tree constructed from a labeled block ordering or other methods may still be large. Since the complexities of our skeleton and complex recovery algorithms are largely dominated by the cardinality of the largest node on the separation tree, it is desirable to further refine our separation tree by reducing the sizes of the nodes. To this end, we propose the following algorithm.

Algorithm 4: Separation Tree Refinement.

Input: A crude separation tree $\mathcal{T}_c$ for $\mathcal{G}$; perfect conditional independence knowledge.
Output: A refined separation tree $\mathcal{T}$ for $\mathcal{G}$.

1 Construct an undirected independence subgraph over each node of $\mathcal{T}_c$;
2 Combine the subgraphs into a global undirected independence graph $\bar{\mathcal{G}}$ whose edge set is the union of all edge sets of subgraphs;
3 Construct a junction tree $\mathcal{T}$ of $\bar{\mathcal{G}}$.

If the current separation tree contains a node whose cardinality is still relatively large, the above algorithm can be repeatedly used until no further refinement is available. However, we remark that the cardinality of the largest node on the separation tree is eventually determined by the sparseness of the underlying chain graph together with other factors including the specific algorithms for constructing undirected independence subgraphs and junction tree. The correctness of the algorithm is proved in Appendix B.

4. Computational Complexity Analysis

In this section, we investigate the computational complexities of Algorithms 1 and 2 and compare them with the pattern recovery algorithm proposed by Studený (1997). We divide our analysis into two stages: (a) skeleton recovery stage (Algorithm 1); (b) complex recovery stage (Algorithm 2). In each stage, we start with a discussion on Studený's algorithm and then give a comprehensive analysis of ours.

4.1 Skeleton Recovery Stage

Let $\mathcal{G} = (V,E)$ be the unknown chain graph, p the number of vertices and e the number of edges, including both lines and arrows. In Studený's algorithm, to delete an edge between a vertex pair u and v, we need to check the independence of u and v conditional on all possible subsets S of $V\backslash\{u,v\}$. Therefore, the complexity for investigating each possible edge in the skeleton is $O(2^p)$ and hence the complexity for constructing the global skeleton is $O(p^2 2^p)$.

For Algorithm 1, suppose that the input separation tree has H nodes $\{C_1,\cdots,C_H\}$ where $H \leq p$ and $m = \max\{\text{card}(C_h), 1 \leq h \leq H\}$. The complexity for investigating all edges within each tree node is thus $O(m^2 2^m)$. Thus, the complexity for the first two parts of Algorithm 1 is $O(Hm^2 2^m)$. For the analysis of the third step, suppose that $k = \max\{\text{card}(S), S \in \mathcal{T}\}$ is the cardinality of the largest separator on the separation tree. Since the tree with H nodes has $H-1$ edges, we also have $H-1$ separators. Thus, the edge to be investigated in step 3 is $O(Hk^2)$. Then, let d be the maximum of the degrees of vertices in the partially recovered skeleton obtained after step 2. By our algorithm,

the complexity for checking each edge is $O(2^d)$. Hence, the total complexity for step 3 is $O(Hk^2 2^d)$. Combining all the three parts, the total complexity for our Algorithm 1 is $O(H(m^2 2^m + k^2 2^d))$. It is usually the case that m, k and d are much smaller than p, and therefore, our Algorithm 1 is computationally less expensive than Studený's algorithm, especially when $\mathcal{G}$ is sparse.

4.2 Complex Recovery Stage

For complex arrow orientation, Studený's algorithm needs to first find a candidate complex structure of some pre-specified length l and then check a collection of conditional independence relations. Finding a candidate structure can be accomplished in a polynomial (of p) time. The complexity of the algorithm is determined by the subsequent investigation on conditional independence relations. Actually, the number of conditional independence relations to be checked for each candidate structure is $2^{p-2} - 1$. Hence, the complexity of Studený's algorithm is exponential in p.

In Algorithm 2, the computational complexity of the large double loop (lines 2-8) is determined by the number of conditional independence relations we check in line 4. After identifying u, w and v, we need only to check one single conditional independence with the conditioning set $S_{uv} \cup \{w\}$, which can be done in $O(1)$ time. The pair $\{u, w\}$ must be connected on $\mathcal{G}'$, thus the number of such pairs is $O(e)$. Since the number of v for each $\{u, w\}$ pair is at most p, we execute line 4 at most $O(pe)$ times. The complexity for the double loop part is thus controlled by $O(pe)$. Next we show that the complexity for taking the pattern of the graph is $O(e^2(p+e))$. As in the remarks on Algorithm 2, in each step, we propose a pair of candidate complex arrows $u_1 \to w_1$ and $u_2 \to w_2$ and then check whether there is an undirected path from w_1 to w_2 whose intermediate vertices are adjacent to neither u_1 nor u_2. This can be done by performing a graph traversal algorithm on an undirected graph which is obtained by deleting all existing arrows on the current graph with the adjacency relations checked on the original graph. By a breadth-first search algorithm, the complexity is $O(p+e)$ (Cormen et al., 2001). Since the number of candidate complex arrow pairs is controlled by $O(e^2)$, the total complexity for finding the pattern is $O(e^2(p+e))$. Since $pe = O(e^2(p+e))$, the total complexity for Algorithm 2 is $O(e^2(p+e))$.

By incorporating the information obtained in the skeleton recovery stage, we greatly reduce the number of conditional independence tests to be checked and hence obtain an algorithm of only polynomial time complexity for the complex recovery stage. The improvement is achieved with additional cost: we need to store the c-separator information S_{uv}'s obtained from Algorithm 1 in the set $\mathcal{S}$. The possible number of $\{u, v\}$ combinations is $O(p^2)$, while the length of S_{uv} is $O(p)$. Hence, the total space we need to store $\mathcal{S}$ is $O(p^3)$, which is still polynomially complex.

4.3 Comparison with a DAG Specific Algorithm When the Underlying Graph is a DAG

In this subsection, we compare our algorithm with Algorithm 1 in Xie et al. (2006) that is designed specifically for DAG structural learning when the underlying graph structure is a DAG. We make this choice of the DAG specific algorithm so that both algorithms can have the same separation tree as input and hence are directly comparable.

Combining the analyses in the above two subsections, we know that the total complexity of our general algorithm is $O(H(m^2 2^m + k^2 2^d) + e^2(p+e))$, while the complexity of the DAG specific algorithm is $O(Hm^2 2^m)$ as claimed in Xie et al. (2006, Section 6). So the extra complexity in the worst case is $O(Hk^2 2^d + e^2(p+e))$. The term that might make a difference is $O(Hk^2 2^d)$, which occurs as the complexity of step 3 in our Algorithm 1 and involves an exponential term in

d. Note that d is defined as the maximum degree of the vertices in the partially recovered skeleton $\mathcal{G}'$ obtained after step 2 of Algorithm 1, where $\mathcal{G}'$ is exactly the skeleton for the underlying DAG in the current situation (i.e., step 3 of Algorithm 1 does not make any further modification to $\mathcal{G}'$ when $\mathcal{G}$ is a DAG). Hence, if the underlying graph is sparse, d is small and the extra complexity $O(Hk^2 2^d + e^2(p+e))$ is well under control.

Therefore, if we believe that the true graph is sparse, the case where our decomposition approach is most applicable, we can apply our general chain graph structural learning algorithm without worrying much about significant extra cost even when the underlying graph is indeed a DAG.

5. Simulation

In this section, we investigate the performance of our algorithms under a variety of circumstances using simulated data sets. We first demonstrate various aspects of our algorithms by running them on randomly generated chain graph models. We then compare our methods with DAG-specific learning algorithms on data generated from the ALARM network, a Bayesian network that has been widely used in evaluating the performance of structural learning algorithms. The simulation results show the competitiveness of our algorithms, especially when the underlying graph is sparse. From now on, we refer to our method as the LCD (Learn Chain graph via Decomposition) method. Algorithms 1 and 2 have been implemented in the R language. All the results reported here are based on the R implementation.

5.1 Performance on Random Chain Graphs

To assess the quality of a learning method, we adopt the way Kalisch and Bühlmann (2007) used in investigating the performance of PC algorithm on Bayesian networks. We perform our algorithms on randomly generated chain graphs and report summary error measures.

5.1.1 DATA GENERATION PROCEDURE

First we discuss the way in which the random chain graphs and random samples are generated. Given a vertex set V, let $p = |V|$ and N denote the average degree of edges (including undirected and pointing out and pointing in) for each vertex. We generate a random chain graph on V as follows:

1. Order the p vertices and initialize a $p \times p$ adjacency matrix A with zeros;

2. For each element in the lower triangle part of A, set it to be a random number generated from a Bernoulli distribution with probability of occurrence $s = N/(p-1)$;

3. Symmetrize A according to its lower triangle;

4. Select an integer k randomly from $\{1, \cdots, p\}$ as the number of chain components;

5. Split the interval $[1, p]$ into k equal-length subintervals $I_1, \cdots, I_k$ so that the set of variables falling into each subinterval I_m forms a chain component C_m;

6. Set $A_{ij} = 0$ for any (i, j) pair such that $i \in I_l, j \in I_m$ with $l > m$.

This procedure then yields an adjacency matrix A for a chain graph with ($A_{ij} = A_{ji} = 1$) representing an undirected edge between V_i and V_j and ($A_{ij} = 1, A_{ji} = 0$) representing a directed edge from V_i to V_j. Moreover, it is not hard to see that $\mathbb{E}[\text{vertex degree}] = N$ where an adjacent vertex can be linked by either an undirected or a directed edge.

Given a randomly generated chain graph $\mathcal{G}$ with ordered chain components $C_1, \cdots, C_k$, we generate a Gaussian distribution on it via the incomplete block-recursive regression as described in Wermuth (1992). Let X_m be the $|C_m| \times 1$ random vector, and $X = (X_k^T, \cdots, X_1^T)^T$. Then we have the block-recursive regression system as

$$B^* X = W^*,$$

where

$$B^* = \begin{pmatrix} \Sigma^{k,k} & \Sigma^{k,k-1} & \cdots & \Sigma^{k,1} \\ 0 & \Sigma^{k-1,k-1.k} & \cdots & \Sigma^{k-1,1.k} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \Sigma^{1,1.23\cdots k} \end{pmatrix}.$$

Let

$$T = \begin{pmatrix} \Sigma^{k,k} & & 0 \\ & \ddots & \\ 0 & & \Sigma^{1,1.23\cdots k} \end{pmatrix}$$

be the block-diagonal part of B^*. Each block diagonal element $\Sigma^{i,i.i+1\cdots k}$ of B^* is the inverse covariance matrix of X_i conditioning on $(X_{i+1}, \cdots, X_k)$, an element of which is set to be zero if the corresponding edge within the chain component is missing. The upper triangular part of B^* has all the conditional covariances between X_i and $(X_1, \cdots, X_{i-1})$. The zero constraints on the elements correspond to missing directed edges among different components. Finally, $W^* \sim N(0, T)$.

For the chain component C_i, suppose the corresponding vertices are $V_{i_1}, \cdots, V_{i_r}$, and in general, let $B^*[V_l, V_m]$ be the element of B^* that corresponds to the vertex pair (V_l, V_m). In our simulation, we generate the B^* matrix in the following way:

1. For the diagonal block $\Sigma^{i,i.i+1\cdots k}$ of B^*, for $1 \le j < j' \le r$, we fix $B^*[V_{i_j}, V_{i_j}] = 1$ and set $B^*[V_{i_j}, V_{i_{j'}}] = 0$ if the vertices V_{i_j} and $V_{i_{j'}}$ are non-adjacent in C_i and otherwise sampled randomly from $(-1.5/r, -0.5/r) \cup (0.5/r, 1.5/r)$, and finally we symmetrize the matrix according to its upper triangular part.

2. For $\Sigma^{i,j.i+1\cdots k}, 1 \le j \le i-1$, an element $B^*[V_l, V_m]$ is set to be zero if $V_l \in C_i$ is not pointed to by an arrow starting from $V_m \in C_1 \cup \cdots \cup C_{i-1}$, and sampled randomly from $(-1.5/r, -0.5/r) \cup (0.5/r, 1.5/r)$ otherwise.

3. If any of the block diagonal elements in B^* is not positive semi-definite, we repeat Step (1) and (2).

After setting up the B^* matrix, we take its block diagonal to obtain the T matrix. For fixed B^* and T, we first draw i.i.d. samples of W^* from $N(0, T)$, and then pre-multiply them by $(B^*)^{-1}$ to get random samples of X. We remark that faithfulness is not necessarily guaranteed by the current sampling procedure and quantifying the deviation from the faithfulness assumption is beyond the scope of this paper.

5.1.2 PERFORMANCE UNDER DIFFERENT SETTINGS

We examine the performance of our algorithm in terms of three error measures: (a) the true positive rate (TPR) and (b) the false positive rate (FPR) for the skeleton and (c) the structural Hamming distance (SHD) for the pattern. In short, TPR is the ratio of # (correctly identified edge) over total number of edges, FPR is the ratio of # (incorrectly identified edge) over total number of gaps and SHD is the number of legitimate operations needed to change the current pattern to the true one, where legitimate operations are: (a) add or delete an edge and (b) insert, delete or reverse an edge orientation. In principle, a large TPR, a small FPR and a small SHD indicate good performance.

In our simulation, we change three parameters p (the number of vertices), n (sample size) and N (expected number of adjacent vertices) as follows:

- $p \in \{10, 40, 80\}$,
- $n \in \{100, 300, 1000, 3000, 10000, 30000\}$,
- $N \in \{2, 5\}$.

For each (p, N) combination, we first generate 25 random chain graphs. We then generate a random Gaussian distribution based on each graph and draw an identically independently distributed (i.i.d.) sample of size n from this distribution for each possible n. For each sample, three different significance levels ($\alpha = 0.005,\ 0.01$ or 0.05) are used to perform the hypothesis tests. We then compare the results to access the influence of the significance testing level on the performance of our algorithms. A separation tree is obtained through the following 'one step elimination' procedure:

1. We start from a complete UIG over all p vertices;
2. We test zero partial correlation for each element of the sample concentration matrix at the chosen significance level α and delete an edge if the corresponding test doesn't reject the null hypothesis;
3. An UIG is obtained after Step 2 and its junction tree is computed and used as the separation tree in the algorithms.

The plots of the error measures are given in Fig. 8, 9 and 10. From the plots, we see that: (a) our algorithms yield better results on sparse graphs ($N = 2$) than on dense graphs ($N = 5$); (b) the TPR increases with sample size while the SHD decreases; (c) the behavior of FPR is largely regulated by the significance level α used in the individual tests and has no clear dependence on the sample size (Note that FPRs and their variations in the middle columns of Fig. 8, 9 and 10 are very small since the vertical axes have very small scales); (d) large significance level α(=0.05) typically yields large TPR, FPR and SHD while the advantage in terms of a larger TPR (compared to $\alpha = 0.005$ or 0.01) fades out as the sample size increases and the disadvantage in terms of a larger SHD becomes much worse; (e) accuracy in terms of TPR and SHD based on $\alpha = 0.005$ or $\alpha = 0.01$ is very close while choosing $\alpha = 0.005$ does yield a consistently (albeit slightly) lower FPR across all the settings in the current simulation. Such empirical evidence suggests that in order to account for the multiple testing problem, we can choose a small value (say $\alpha = 0.005$ or 0.01 for the current example) for the significance level of individual tests. However, the optimal value for a desired overall error rate may depend on the sample size and the sparsity of the underlying graph.

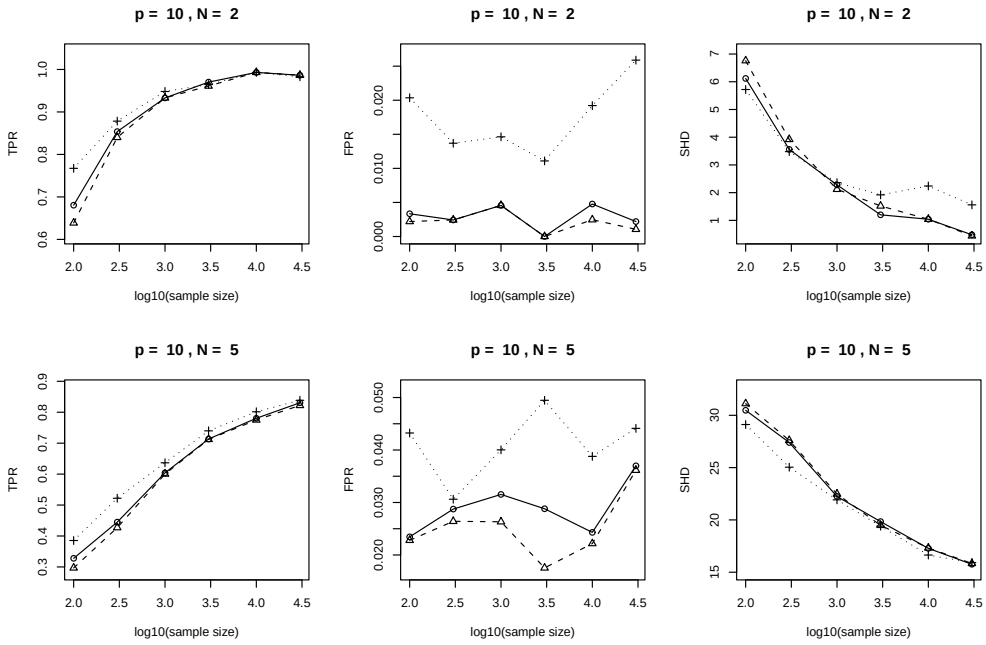


Figure 8: Error measures of the algorithms for randomly generated Gaussian chain graph models: average over 25 repetitions with 10 variables. The two rows correspond to $N = 2$ and $N = 5$ cases and the three columns give three error measures: TPR, FPR and SHD in each setting respectively. In each plot, the solid/dashed/dotted lines correspond to significance levels $\alpha = 0.01/0.005/0.05$.

Finally, we look at how our method scales with the sample size, which is not analyzed explicitly in Section 4. The average running times vs. the sample sizes are plotted in Fig. 11. It can be seen that: (a) the average run time scales approximately linearly with $\log(\text{sample size})$; and (b) the scaling constant depends on the sparsity of the graph. The simulations were run on an Intel Core Duo 1.83GHz CPU.

5.2 Learning the ALARM Network

As we have pointed out in Section 1, Bayesian networks are special cases of chain graphs. It is of interest to see whether our general algorithms still work well when the data are actually generated from a Bayesian network. For this purpose, in this subsection, we perform simulation studies on both Gaussian and discrete case for the ALARM network in Fig. 12 and compare our algorithms for general chain graphs with those specifically designed for Bayesian networks. The network was first proposed in Beinlich et al. (1989) as a medical diagnostic network.

5.2.1 THE GAUSSIAN CASE

In the Gaussian case, for each run of the simulation, we repeat the following steps:

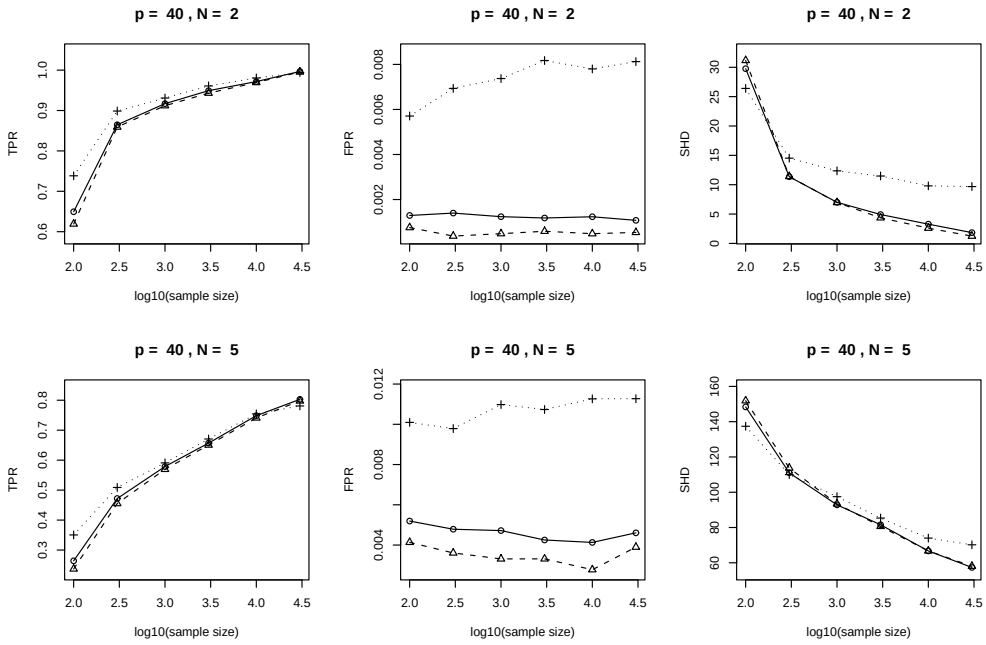


Figure 9: Error measures of the algorithms for randomly generated Gaussian chain graph models: average over 25 repetitions with 40 variables. Display setup is the same as in Fig. 8.

1. A Gaussian distribution on the network is generated using a recursive linear regression model, whose coefficients are random samples from the uniform distribution on $(-1.5,-0.5)\cup(0.5,1.5)$ and residuals are random samples from $N(0,1)$.

2. A sample of size n is generated from the distribution obtained at step 1.

3. We run the LCD algorithms, the DAG learning algorithms proposed in Xie et al. (2006) and the PC algorithm implemented in the R package `pcalg` (Kalisch and Bühlmann, 2007) all with several different choices of the significance level α. The one step elimination procedure described in Section 5.1.2 was used to construct the separation trees for the LCD and DAG methods.

4. We record the number of extra edges (FP), the number of missing edges (FN) and the structural Hamming distance (SHD) compared with the true pattern for all the three learned patterns.

We performed 100 runs for each sample size $n \in \{1000, 2000, 5000, 10000\}$. For each sample, we allow three different significance levels $\alpha \in \{0.05, 0.01, 0.005\}$ for all the three methods. Table 1 documents the averages and standard errors (in parentheses) from the 100 runs for each method-parameter-sample size combination.

As shown in Table 1, compared with the DAG method (Xie et al., 2006), the LCD method consistently yields a smaller number of false positives and the differences in false negatives are consistently smaller than two on recovering the skeleton of the network. The SHDs obtained from

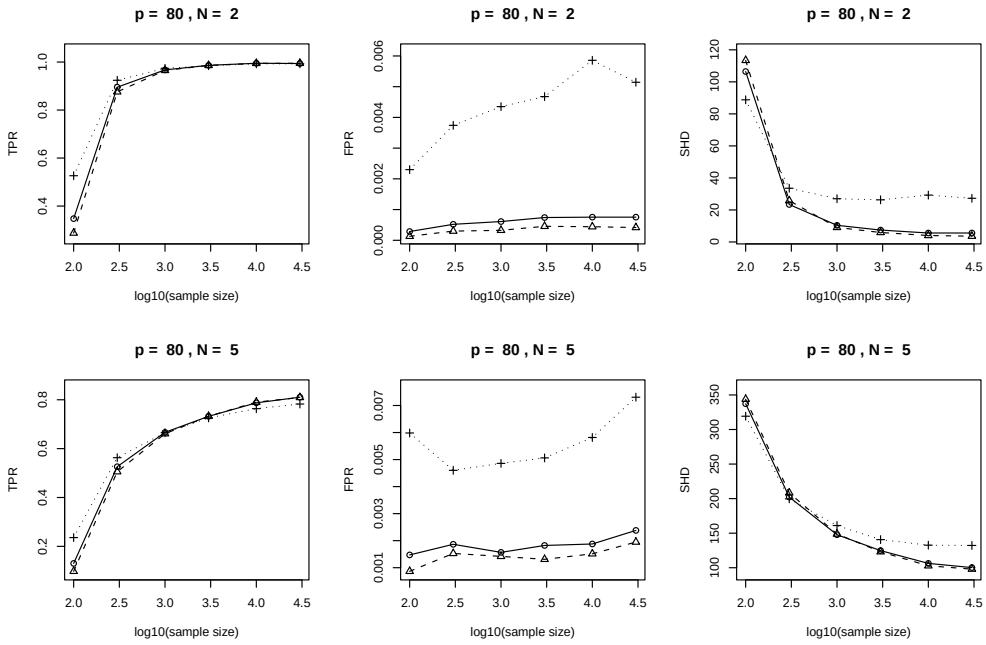


Figure 10: Error measures of the algorithms for randomly generated Gaussian chain graph models: average over 25 repetitions with 80 variables. Display setup is the same as in Fig. 8.

our algorithms are usually comparable to those from the DAG method when $\alpha = 0.05$ and the difference is usually less than five when $\alpha = 0.01$ or 0.005. Moreover, we remark that as sample size grows, the power of the significance test increases, which leads to better performance of the LCD method as in the case of other hypothesis testing based methods. However, from Table 1, we find that the LCD performance increases more rapidly in terms of SHD. One plausible reason is that in Algorithm 2, we identify complex arrows by rejecting conditional independence hypotheses rather than direct manipulation as in the algorithms specific for DAG and hence have some extra benefit in terms of accuracy as the power of the test becomes greater.

Finally, the LCD method consistently outperforms the PC algorithm in all three error measures. Such simulation results confirm that our method is reliable when we do not know the information that the underlying graph is a DAG, which is usually untestable from data.

5.2.2 THE DISCRETE CASE

In this section, a similar simulation study with discrete data sampled from the ALARM network is performed. The variables in the network are allowed to have two to four levels. For each run of the simulation, we repeat the following steps:

1. For each variable X_i and fixed configuration pa_i of its parents, we define the conditional probability $P(X_i = j|pa_i) = r_j / \sum_{k=1}^{L} r_k$, where L is the number of levels of X_i and $\{r_1, \cdots, r_L\}$ are random numbers from the Uniform(0,1) distribution.

2. A sample of size n is generated from the above distribution.

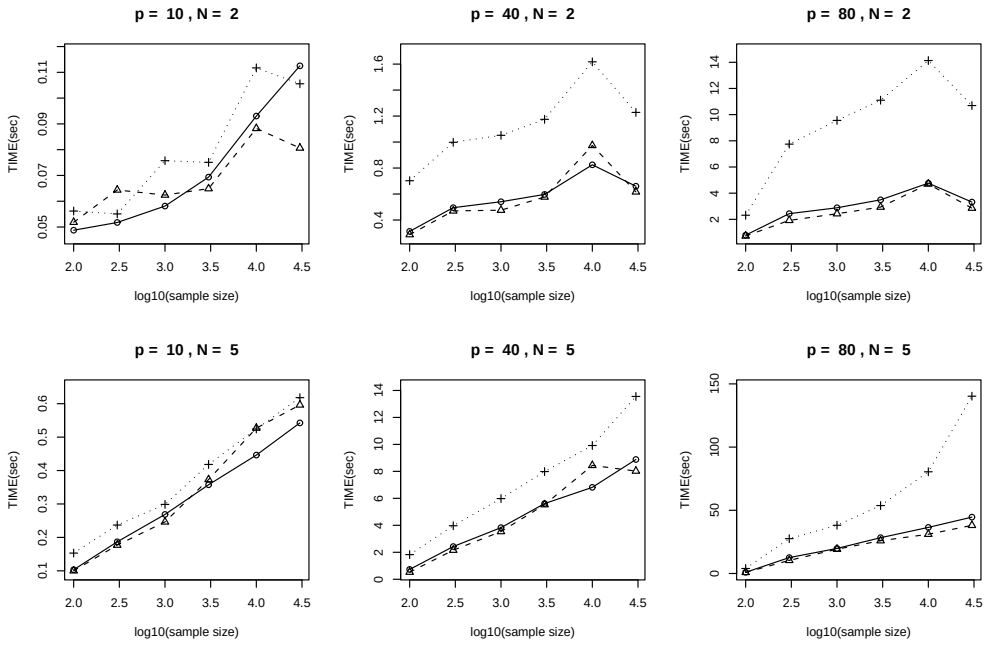


Figure 11: Running times of the algorithms on randomly generated Gaussian chain graph models: average over 25 repetitions. The two rows correspond to $N = 2$ and 5 cases and the three columns represent $p = 10,\ 40$ and 80 respectively. In each plot, the solid/dashed/dotted lines correspond to significance levels $\alpha = 0.01/0.005/0.05$.

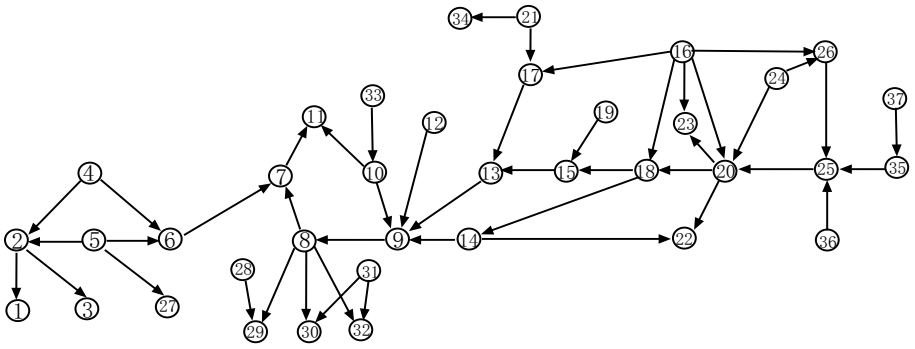


Figure 12: The ALARM network.

3. We run three algorithms designed specifically for DAG that have been shown to have a good performance: MMHC (Max-Min Hill Climbing: Tsamardinos et al., 2006), REC (Recursive: Xie and Geng, 2008) and SC (Sparse Candidate: Friedman et al., 1999) with several different choice of parameters. We also perform the LCD learning algorithm with different choices of the significance level. For the LCD method, a grow-shrink Markov blanket selection is

Alg (Level α)	$n = 1000$	$n = 2000$	$n = 5000$	$n = 10000$
DAG	(3.09, 4.05, 17.3)	(3.17, 3.14, 15.2)	(3.48, 2.07, 12.4)	(3.16, 1.62, 10.5)
(0.05)	(0.18, 0.19, 0.61)	(0.16, 0.16, 0.54)	(0.18, 0.16, 0.55)	(0.17, 0.11, 0.46)
DAG	(0.87, 3.41, 12.2)	(0.87, 2.60, 9.90)	(0.71, 1.60, 6.02)	(0.58, 1.24, 5.23)
(0.01)	(0.08, 0.19, 0.64)	(0.09, 0.17, 0.54)	(0.08, 0.14, 0.41)	(0.08, 0.10, 0.36)
DAG	(0.61, 3.34, 11.4)	(0.54, 2.50, 8.77)	(0.36, 1.49, 5.43)	(0.33, 1.08, 4.14)
(0.005)	(0.08, 0.19, 0.60)	(0.07, 0.16, 0.52)	(0.07, 0.12, 0.38)	(0.06, 0.11, 0.39)
LCD	(2.06, 5.19, 19.0)	(2.10, 4.25, 16.7)	(2.5, 3.07, 14.0)	(2.15, 2.38, 11.3)
(0.05)	(0.16, 0.19, 0.58)	(0.15, 0.17, 0.53)	(0.15, 0.15, 0.50)	(0.15, 0.14, 0.39)
LCD	(0.41, 4.93, 15.8)	(0.34, 4.01, 12.9)	(0.44, 2.82, 9.79)	(0.30, 2.10, 7.11)
(0.01)	(0.06, 0.21, 0.61)	(0.06, 0.17, 0.50)	(0.07, 0.15, 0.41)	(0.05, 0.13, 0.40)
LCD	(0.22, 4.86, 15.3)	(0.12, 3.85, 12.2)	(0.13, 2.85, 9.14)	(0.14, 1.95, 6.49)
(0.005)	(0.05, 0.21, 0.61)	(0.03, 0.16, 0.52)	(0.04, 0.14, 0.41)	(0.03, 0.13, 0.40)
PC	(4.72, 7.98, 38.4)	(5.03, 6.79, 38.5)	(4.94, 5.19, 35.2)	(4.41, 4.21, 31.9)
(0.05)	(0.22, 0.23, 0.73)	(0.23, 0.23, 0.73)	(0.24, 0.20, 0.79)	(0.26, 0.19, 0.89)
PC	(3.45, 9.23, 37.9)	(3.11, 7.78, 34.8)	(3.27, 5.88, 31.5)	(2.98, 4.87, 30.9)
(0.01)	(0.19, 0.26, 0.78)	(0.19, 0.22, 0.76)	(0.20, 0.21, 0.79)	(0.22, 0.20, 0.88)
PC	(3.14, 9.61, 38.1)	(2.95, 8.05, 35.6)	(3.03, 6.15, 31.4)	(2.88, 5.13, 30.4)
(0.005)	(0.20, 0.27, 0.69)	(0.18, 0.23, 0.79)	(0.20, 0.21, 0.78)	(0.22, 0.20, 0.79)

Table 1: Simulation results for Gaussian samples from the ALARM network. Averages and standard errors for (FP, FN, SHD) from 100 runs.

performed on the data to learn the UIG and the junction tree of the UIG is supplied as the separation tree for the algorithm.

4. For each algorithm, we recorded the FP, FN and SHD of the recovered pattern under each choice of the learning parameter.

We performed 100 runs for each of the four different sample sizes $n \in \{1000, 2000, 5000, 10000\}$, and in each run, we allow the following choices of the learning parameters:

Alg (Level α)	$n = 1000$	$n = 2000$	$n = 5000$	$n = 10000$
MMHC	(0.27, 7.77, 34.0)	(0.16, 5.15, 27.9)	(0.08, 2.61, 20.6)	(0.03, 1.53, 16.0)
(0.05)	(0.05, 0.25, 0.57)	(0.04, 0.19, 0.48)	(0.03, 0.15, 0.47)	(0.02, 0.10, 0.45)
MMHC	(0.13, 8.39, 34.6)	(0.08, 5.53, 28.2)	(0.07, 2.96, 21.2)	(0.00, 1.64, 16.1)
(0.01)	(0.04, 0.26, 0.57)	(0.03, 0.19, 0.48)	(0.03, 0.16, 0.46)	(0.00, 0.10, 0.45)
MMHC	(0.13, 8.79, 35.3)	(0.09, 5.77, 28.6)	(0.06, 3.09, 21.4)	(0.01, 1.75, 16.3)
(0.005)	(0.04, 0.26, 0.57)	(0.03, 0.18, 0.51)	(0.03, 0.16, 0.47)	(0.01, 0.11, 0.46)
REC	(6.20, 4.95, 40.1)	(6.49, 3.52, 35.9)	(6.20, 1.77, 28.9)	(6.39, 0.80, 25.1)
(0.05)	(0.24, 0.19, 0.71)	(0.24, 0.13, 0.74)	(0.23, 0.11, 0.68)	(0.23, 0.08, 0.73)
REC	(1.57, 5.52, 33.2)	(1.82, 3.64, 27.2)	(2.06, 1.80, 20.0)	(2.31, 0.86, 16.2)
(0.01)	(0.13, 0.22, 0.58)	(0.13, 0.14, 0.48)	(0.12, 0.11, 0.56)	(0.14, 0.09, 0.52)
REC	(1.02, 5.72, 32.9)	(1.23, 3.76, 26.3)	(1.20, 1.86, 18.6)	(1.64, 0.90, 14.6)
(0.005)	(0.10, 0.22, 0.55)	(0.10, 0.15, 0.49)	(0.08, 0.12, 0.49)	(0.12, 0.09, 0.47)
SC	(0.63, 7.35, 34.2)	(0.50, 4.71, 27.8)	(0.59, 2.50, 21.7)	(0.81, 1.53, 18.1)
(5)	(0.07, 0.26, 0.60)	(0.09, 0.18, 0.49)	(0.09, 0.14, 0.53)	(0.11, 0.10, 0.56)
SC	(0.85, 7.30, 34.5)	(0.76, 4.65, 28.3)	(0.94, 2.36, 22.0)	(1.30, 1.32, 18.1)
(10)	(0.09, 0.25, 0.64)	(0.09, 0.18, 0.52)	(0.09, 0.14, 0.53)	(0.13, 0.10, 0.59)
LCD	(2.92, 8.49, 38.9)	(2.50, 5.94, 32.17)	(2.18, 3.41, 25.17)	(1.99, 2.18, 19.8)
(0.05)	(0.17, 0.21, 0.53)	(0.16, 0.18, 0.51)	(0.14, 0.13, 0.46)	(0.12, 0.10, 0.50)
LCD	(1.07, 8.16, 37.8)	(0.97, 5.63, 31.7)	(0.80, 3.40, 23.1)	(0.68, 2.11, 17.2)
(0.01)	(0.09, 0.20, 0.46)	(0.09, 0.16, 0.42)	(0.09, 0.13, 0.46)	(0.09, 0.09, 0.42)
LCD	(0.69, 8.14, 38.4)	(0.67, 5.84, 31.8)	(0.40, 3.31, 23.2)	(0.41, 2.09, 17.1)
(0.005)	(0.08, 0.19, 0.41)	(0.08, 0.17, 0.43)	(0.07, 0.13, 0.45)	(0.07, 0.09, 0.40)

Table 2: Simulation results for discrete samples from the ALARM network. Averages and standard errors for (FP, FN, SHD) from 100 runs.

- For MMHC, REC and LCD, we allow three different significance levels $\alpha \in \{0.05, 0.01, 0.005\}$; and
- For SC, we allow the learning parameters to be either 5 or 10.

The means and standard errors from the 100 runs of all the four learning methods are summarized in Table 2. It can be seen that REC could yield the best result when the learning parameter is appropriately chosen. However, all the other methods are more robust against the choice of learning parameters. For the LCD method, all the three error measures: FP, FN and SHD are consistently comparable to those of methods specifically designed for DAG. Moreover, as in the Gaussian case, the power of the tests used in the LCD method grows as the sample size become larger, which makes the LCD method even more competitive, especially in terms of SHD. For example, when the sample size reaches 10000, the LCD method with $\alpha = 0.01$ or 0.005 outperforms the sparse candidate method with parameters 5 or 10.

6. Discussion

In this paper, we presented a computationally feasible method for structural learning of chain graph models. The method can be used to facilitate the investigation of both response-explanatory and symmetric association relations among a set of variables simultaneously within the framework of chain graph models, a merit not shared by either Bayesian networks or Markov networks.

Simulation studies illustrate that our method yields good results in a variety of situations, especially when the underlying graph is sparse. On the other hand, the results also reveal that the power of the significance test has an important influence on the performance of our method. With fixed number of samples, one can expect a better accuracy if we replace the asymptotic test used in our implementation with an exact test. However, there is a trade-off between accuracy and computational time.

The results in this paper also raised a number of interesting questions for future research. We briefly comment on some of those questions here. First, the separation tree plays a key role in Algorithms 1 and 2. Although the construction of separation trees has been discussed in Xie et al. (2006) and Section 3.5 here, we believe that there is room for further improvements. Second, we have applied hypothesis testing for the detection of local separators in Algorithm 1 and also in complex arrow determination in Algorithm 2. It shall be interesting to see whether there exists some alternative approach, preferably not based on hypothesis testing, to serve the same purpose here. A theoretical analysis of the effect of multiple testing on the overall error rate of the procedure is also important. In addition, it is a common practice to incorporate prior information about the order of the variables in graphical modelling. Therefore, incorporation of such information into our algorithms is worth investigation. Finally, our approach might be extendible to the structural learning of chain graph of alternative Markov properties, for example, AMP chain graphs (Andersson et al., 2001) and multiple regression chain graphs (Cox and Wermuth, 1996).

An R language package `lcd` that implements our algorithms is available on the first author's website: `www.stanford.edu/~zongming/software.html`.

Acknowledgments

The authors would like to thank two referees for their valuable suggestions and comments which improve the presentation of the previous version of the paper. We would also like to thank Professor John Chambers and Xiangrui Meng for their help on the simulation study. This research was supported by NSFC (10771007, 10431010, 10721403), 863 Project of China (2007AA01Z437), MSRA and MOE-Microsoft Key Laboratory of Statistics and Information Technology of Peking University. The first author was also supported in part by grants NSF DMS 0505303 and NIH EB R01 EB001988.

Appendix A. Proofs of Theoretical Results

In this part, we give proofs to theorems and propositions. We first give a definition and several lemmas that are to be used in later proofs.

Definition 7 *Let $\mathcal{T}$ be a separation tree for a CG $\mathcal{G}$ with the node set $\mathcal{C} = \{C_1, \cdots, C_H\}$. For any two vertices u and v in $\mathcal{G}$, the* distance between u and v in the tree $\mathcal{T}$ *is defined by*

$$d(u,v) = \min_{C_i \ni u, C_j \ni v} d(C_i, C_j),$$

where $d(C_i, C_j)$ is the distance between nodes C_i and C_j in $\mathcal{T}$. We call C_i and C_j minimizers *for u and v if they minimize the distance $d(C_i, C_j)$.*

Lemma 8 *Let ρ be a route from u to v in a chain graph $\mathcal{G}$, and W the set of all vertices on ρ (W may or may not contain the two end vertices). Suppose that ρ is intervented by $S \subset V$. If $W \subset S$, ρ is also intervented by W and any vertex set containing W.*

Proof Since ρ is intervented by S and $W \subset S$, there must be a non head-to-head section σ of ρ that is hit by S and actually every non head-to-head section of ρ is hit by S. Thus, σ is also hit by W and any vertex set containing W. Hence, ρ is intervented. ■

Lemma 9 *Let $\mathcal{T}$ be a separation tree for a chain graph over vertex set V and K a separator of $\mathcal{T}$ which separates $\mathcal{T}$ into two subtrees $\mathcal{T}_1$ and $\mathcal{T}_2$ with variable sets V_1 and V_2. Suppose that $u \in V_1 \backslash K$, $v \in V_2 \backslash K$ and ρ is a route from u to v in $\mathcal{G}$. Let W denote the set of all vertices on ρ (W may or may not contain the two end vertices). Then ρ is intervented by $W \cap K$ and by any vertex set containing $W \cap K$.*

Proof Since $u \in V_1 \backslash K$ and $v \in V_2 \backslash K$, there must be a sequence from s (may be u) to y (may be v) in $\rho = (u, \cdots, s, t, \cdots, x, y, \cdots, v)$ such that $s \in V_1 \backslash K$, $y \in V_2 \backslash K$ and all vertices from t to x in this sequence are contained in K. Otherwise, every vertex of ρ is either in $V_1 \backslash K$ or in $V_2 \backslash K$. This implies that there exists $w \in V_1 \backslash K$ and $z \in V_2 \backslash K$ on ρ that are adjacent, which is contradictory to the fact that $\langle V_1 \backslash K, V_2 \backslash K | K \rangle_{\mathcal{G}}^{sep}$. Without loss of generality, we can suppose that y is the first vertex (from the left) of ρ that is not in V_1.

Let ρ' be the sub-route of the sequence $(s, t, \cdots, x, y)$, and W' be the vertex set of ρ' excluding s and y. Since $W' \subset K$, we know from Lemma 8 that there is at least one non head-to-head section

(w.r.t. ρ') on ρ' and every non head-to-head section of ρ' is hit by W'. We are to show that there is at least one non head-to-head section of ρ that is hit by K and hence $W \cap K$ as well as any set containing $W \cap K$.

The only problem arises when ρ' is part of a head-to-head section of ρ. Otherwise, there is some non head-to-head section of ρ' that is (part of) a non head-to-head section of ρ.

Thus, we suppose that the head-to-head section of ρ is

$$s' \to s" - \cdots - s - t - \cdots - x - y - \cdots - y" \leftarrow y'.$$

By our assumption on y, we know that $s' \in V_1$. If $s' \in K$, then the non head-to-head section containing s' is hit by K. If $s' \in V_1 \backslash K$ and $y' \in K$, then the non head-to-head section containing y' gives the result. If $s' \in V_1 \backslash K$ and $y' \in V_1 \backslash K$, then we can consider the sub-route starting from y'. This is legitimate since every non head-to-head section of that sub-route is also non head-to-head w.r.t. ρ. Hence, we need only consider the case that $s' \in V_1 \backslash K$ and $y' \in V_2 \backslash K$. In this case, let t' be the last (from left) vertex in this section that is adjacent to s' and x' the first vertex after t' in this section that is adjacent to y'. Since chain graphs cannot have directed pseudocycles, we know that $s' \to t'$ and $y' \to x'$. Then we have $s' \not\perp\!\!\!\perp y' | K$, which is contradictory to the property of separation trees that $\langle V_1 \backslash K, V_2 \backslash K | K \rangle_{\mathcal{G}}^{sep}$. This completes our proof. ■

Lemma 10 *Let u and v be two non adjacent vertices in a chain graph $\mathcal{G}$ and ρ a route from u to v in $\mathcal{T}$. If ρ is not contained in $\mathrm{An}(u) \cup \mathrm{An}(v)$, then ρ is intervented by any subset S of $\mathrm{An}(u) \cup \mathrm{An}(v)$.*

Proof Since ρ is not contained in $\mathrm{An}(u) \cup \mathrm{An}(v)$, there exist four vertices s, t, x and y, such that $\rho = (u, \cdots, s, t, \cdots, x, y, \cdots, v)$, with $\{s, y\} \subset \mathrm{An}(u) \cup \mathrm{An}(v)$ and $\{t, \cdots, x\} \cap [\mathrm{An}(u) \cup \mathrm{An}(v)] = \emptyset$. Then we have $s \to t$ and $x \leftarrow y$, since otherwise t and/or x must be in $\mathrm{An}(u) \cup \mathrm{An}(v)$. Thus, there exists at least one head-to-head section between s and y on ρ such that it is not hit by any subset of $\mathrm{An}(u) \cup \mathrm{An}(v)$. Hence, ρ is intervented by any subset S of $\mathrm{An}(u) \cup \mathrm{An}(v)$. ■

Lemma 11 *Let $\mathcal{T}$ be a separation tree for a chain graph $\mathcal{G}$ over V and C a node of $\mathcal{T}$. Let u and v be two vertices in C which are non adjacent in $\mathcal{G}$. If u and v are not contained simultaneously in any separator connected to C, then there exists a subset S of C which c-separates u and v in $\mathcal{G}$.*

Proof Define

$$S = [\mathrm{An}(u) \cup \mathrm{An}(v)] \cap [C \backslash \{u, v\}].$$

We show below that $\langle u, v | S \rangle_{\mathcal{G}}^{sep}$.

To this end, let ρ be any fixed route from u to v in $\mathcal{G}$. If ρ is not contained in $\mathrm{An}(u) \cup \mathrm{An}(v)$, by Lemma 10, ρ is intervented by S. Otherwise, we divide the problem into the following six possible situations:

1. $u - \cdots - u' \leftarrow x$, $x \neq v$, $x \in C$, where $u - \cdots - u'$ means the first (from left) section of ρ that contains u;
2. $u - \cdots - u' \to x - \cdots - x' \to y$, $x \neq v$, $x \in C$;

3. $u - \cdots - u' \to x - \cdots - x' \leftarrow y$, $x \neq v$, $x \in C$, $y \in C$;

4. $u - \cdots - u' \to x - \cdots - x' \leftarrow y$, $x \neq v$, $x \in C$, $y \notin C$;

5. $u - u' - \cdots - v' - v$, $u - u' - \cdots - v' \to v$ or $u - u' - \cdots - v' \leftarrow v$;

6. $u - \cdots - u' \to x$ or $u - \cdots - u' \leftarrow x$, $x \notin C$.

We prove the desired result situation by situation.

For situation 1, we have that $x \in \mathrm{An}(u)$, which, together with $x \in C$ implies that $x \in S$. The non head-to-head section containing x is hit by S, and ρ is thus intervented.

For situation 2, since $x \in \mathrm{An}(u) \cup \mathrm{An}(v)$ and $x \notin \mathrm{An}(u)$, we have $x \in \mathrm{An}(v)$. Together with $x \in C$, this gives $x \in S$ and the non head-to-head section containing x is hit by S.

For situation 3, since chain graphs do not admit directed pseudocycles, we know that $x \notin \mathrm{An}(u)$ and $y \neq v$. Similar to situation 2, we have $x \in \mathrm{An}(v)$ and hence $y \in \mathrm{An}(v)$. The non head-to-head section containing y is hit by S.

For situation 4, suppose that C' is one of the nodes on $\mathcal{T}$ that contains y. Consider first the case when v belongs to the separator K connected to C and the next node on the path from C to C' on $\mathcal{T}$. By our assumption, $u \notin K$. We divide the problem into the following three cases:

(i) $\{u, \cdots, u'\} \cap S \neq \emptyset$: $u - \cdots - u'$ is hit by S and ρ is hence intervented;

(ii) $\{u, \cdots, u'\} \cap S = \emptyset, \{x, \cdots, x'\} \cap S = \emptyset$: the head-to-head section $x - \cdots - x'$ is not hit by S and ρ is intervented;

(iii) $\{u, \cdots, u'\} \cap S = \emptyset, \{x, \cdots, x'\} \cap S \neq \emptyset$: there must exist some $x^* \in \{x, \cdots, x'\}$ such that $x^* \in C \cap \mathrm{An}(v)$ and $x^* \neq v$. Since $\{u, \cdots, u'\} \cap S = \emptyset$ and $u \perp\!\!\!\perp y | K$, there should be no complex in the induced subgraph of $(u, \cdots, u', x, \cdots, x', y)$. Otherwise, there exists some $u^* \in \{u, \cdots, u'\}$, such that (u^*, y) is an edge on $\left(\mathcal{G}_{\mathrm{An}(u,y,K)}\right)^m$, which implies $u \not\perp\!\!\!\perp y | K$ since $\{u, \cdots, u'\} \cap K = \emptyset$. However, this requires that there is some $u^{**} \in \{u, \cdots, u'\}$ such that (u^{**}, y) is an edge on $\mathcal{G}$, which again implies that $u \not\perp\!\!\!\perp y | K$. Hence, this case can never happen.

Next, we consider the case that $v \notin K$. The assumption that $\{x, \cdots, x'\} \subset \mathrm{An}(v)$ implies that there exists at least one $' \to'$ on the sub-route l' of ρ from y to v. Consider the rightmost one of such arrows, there is no further $' \leftarrow'$ closer to the right end v than it is. Otherwise, any vertex w between them satisfies $w \in \mathrm{An}(u)$ and $w \in \mathrm{de}(v)$, which is contradictory to the fact that $v \in \mathrm{de}(u)$ here. Thus, this case reduces to one of the situations 1, 5 and 6 with u replaced by v.

For situation 5, since u and v are non adjacent, we know that $v' \neq u$ and $u' \neq v$. If $\{u', \cdots, v'\} \cap S = \emptyset$, then $\{u', \cdots, v'\} \cap C \subset \{u, v\}$. We can eliminate vertices from the left such that $\{u', \cdots, v'\} \cap C \subset \{v\}$. This will not influence our result since any non head-to-head section of the sub-route is (part of) a non head-to-head section of ρ. Since $u' \neq v$ and $\{u', \cdots, v'\} \cap C \subset \{v\}$, we have $u' \notin C$. Suppose that $u' \in C'$ and K is the separator related to C and the next node on the path from C to C' on $\mathcal{T}$. Then $u \in K, v \notin K$ and $u' \perp\!\!\!\perp v | K$. However, since $\{u', \cdots, v'\} \cap C \subset \{v\}$, we have $\{u, \cdots, u'\} \cap K = \emptyset$, which is impossible. Hence $\{u', \cdots, v'\} \cap S \neq \emptyset$ and ρ is intervented.

For situation 6, consider first the case that $u - \cdots - u' \leftarrow x$. If $\{u, \cdots, u'\} \cap S \neq \emptyset$, then ρ is intervented by S. Otherwise, suppose that $x \in C'$ and K is the separator connected to C and the next node on the path from C to C' in $\mathcal{T}$. If $v \in K$, then $u \notin K$ and $\{u, \cdots, u'\} \cap K \subset \{v\}$. If $\{u, \cdots, u'\} \cap K = \{v\}$, then it reduces to situation 5. If $\{u, \cdots, u'\} \cap K = \emptyset$, then $u \not\perp\!\!\!\perp x | K$, which is

contradictory to the definition of separation tree. If $v \notin K$, then by the above argument, we must have $u \in K$. Consider the sub-route of ρ starting with x. It is legitimate to do so since any non head-to-head section of the sub-route is also non head-to-head in ρ. By Lemma 9, at least one non head-to-head section is hit by $W \cap K$ where W is the vertex set of the sub-route excluding the two end vertices. We know that $W \cap K \subset S \cup \{u,v\}$. If the non head-to-head section is hit at u or v, we can consider the further sub-route starting at that point and it is again legitimate by the same reason. Finally, we can reduce to the case where $W \cap K \subset S$. Thus, ρ is intervented by S. This also completes the proof of situation 4. For the other case in this situation, all the argument is the same up to the point where $v \notin K$ and $u \in K$. We can consider reversing the vertex sequence, then with u replaced by v, it must be in one of the situations 1 to 4, the second case in situation 6 or the first case in situation 6 with $u \notin K$. This complete the proof of the lemma. ■

Proof of Theorem 3. The sufficiency of condition 1 is given by Lemma 9. The sufficiencies of conditions 2 and 3 are trivial by the definition of c-separation.

Now we show the necessity part of the theorem. If $d(u,v) > 0$, by Lemma 9, any separator K on the path from minimizers C_i to C_j c-separates u and v. If $d(u,v) = 0$, we consider the following two possible cases: (1) u and v are not contained simultaneously in any separator connected to C for some node C on $\mathcal{T}$ containing both u and v; (2) otherwise. For the first case, Lemma 11 shows that there exists some $S' \subset C$ that c-separates u and v. Otherwise, since $\langle u,v|\mathrm{bd}_{\mathcal{G}}(u) \cup \mathrm{bd}_{\mathcal{G}}(v)\rangle_{\mathcal{G}}^{sep}$, $\mathrm{bd}_{\mathcal{G}}(u) \subset \bigcup_{u\in C} C$ and $\mathrm{bd}_{\mathcal{G}}(v) \subset \bigcup_{v\in C} C$, we know that at least one of the conditions 2 and 3 holds.

Proof of Proposition 4. We verify Proposition 4 by contradiction. Let us suppose that u and v are parents of a complex $\kappa = (u, w_1, \cdots, w_k, v), k \geq 1$ in $\mathcal{T}$ and that for any node C on $\mathcal{T}$, $\{u,v\} \cap C \neq \{u,v\}$. Now suppose that $u \in C_1$, $v \in C_2$ and K is the separator related to C_1 and the next node on the path from C_1 to C_2 on $\mathcal{T}$. If $u \notin K$ and $v \notin K$, we must have that $\{w_1, \cdots, w_k\} \cap K \neq \emptyset$. This implies that $u \not\perp\!\!\!\perp v|K$, which is contrary to the definition of separation trees. Hence, without loss of generality, we may suppose that $u \in K$, and this enables us to go one node closer to C_2 on the path. Then after finite steps, we will consider two adjacent nodes on $\mathcal{T}$. Repeating the above argument ensures that $\{u,v\}$ belongs to one of these two nodes.

Appendix B. Proofs for Correctness of the Algorithms

Before proving the correctness of the algorithms, we need several more lemmas.

Lemma 12 *Suppose that u and v are two adjacent vertices in $\mathcal{G}$, then for any separation tree $\mathcal{T}$ for $\mathcal{G}$, there exists a node C in $\mathcal{T}$ which contains both u and v.*

Proof If not, then there exists a separator K on $\mathcal{T}$, such that $u \in V_1 \backslash K$ and $v \in V_2 \backslash K$ where V_i denotes the variable set of the subtree $\mathcal{T}_i$ induced by removing the edge attached by the separator S, for $i = 1$ and 2. This implies $u \perp\!\!\!\perp v|K$, which is impossible. ■

Lemma 13 *Any arrow oriented in line 5 of Algorithm 2 is correct in the sense that it is an arrow with the same orientation in $\mathcal{G}$.*

Proof We prove the lemma by induction. If we don't orient any arrow in line 5, then the lemma holds trivially. Otherwise, suppose $u \to w$ is the first arrow we orient by considering the ordered triple $\langle u, v, w \rangle$, then we show that it cannot be $u - w$ or $u \leftarrow w$ in $\mathcal{G}$. If it is $u - w$ in $\mathcal{G}$, then by Lemma 11, if u and v are not in any separator simultaneously, there exists some $S_{uv} \subset C_h$ such that $u \perp\!\!\!\perp v | S_{uv}$ and $w \in S_{uv}$. Otherwise, $u \perp\!\!\!\perp v | \mathrm{bd}_{\mathcal{G}}(u) \cup \mathrm{bd}_{\mathcal{G}}(v)$, and we know that $w \in \mathrm{bd}_{\mathcal{G}}(u) \cup \mathrm{bd}_{\mathcal{G}}(v) \subset \mathrm{bd}_{\mathcal{G}'}(u) \cup \mathrm{bd}_{\mathcal{G}'}(v)$. Thus we won't orient it as $u \to w$. A similar argument holds for the case when $u \leftarrow w$ in $\mathcal{G}$.

Now suppose that the k-th arrow we orient is correct, let's consider the $k+1$-th. Suppose it's $u' \to w'$ by considering the order triple $\langle u', v', w' \rangle$. Then the above argument holds exactly with u, v and w substituted by u', v' and w'. However, for here, the claim that $\mathrm{bd}_{\mathcal{G}}(u') \cup \mathrm{bd}_{\mathcal{G}}(v') \subset \mathrm{bd}_{\mathcal{G}'}(u') \cup \mathrm{bd}_{\mathcal{G}'}(v')$ holds by the induction assumption. ■

Lemma 14 *Suppose that $\mathcal{H}$ is a graph, if we disorient any non-complex arrow in $\mathcal{H}$, the pattern of $\mathcal{H}$ does not change.*

Proof First, we note that we will not add or delete edge in $\mathcal{H}$, so the skeleton of $\mathcal{H}$ does not change.

Second, we only disorient non-complex arrows, and hence those complexes in $\mathcal{H}$ before disorientation remain complexes after disorientation since the subgraph induced by any complex does not change.

Finally, we show that there will not be new complex. If there appears a new complex, say $u \to w_1 - \cdots, -w_l \leftarrow v$, we must have $l \geq 2$. Since it was not a complex before disorientation, one of the lines in $w_1 - \cdots - w_l$ must be the arrow disoriented. Suppose we have $w_i \to w_{i+1}$ before disorientation, then $w_i \to w_{i+1} - \cdots - w_l \leftarrow v$ was a complex before disorientation. Hence, the disoriented arrow $w_i \to w_{i+1}$ was a complex arrow, which contradicts our assumption. ■

Correctness of Algorithm 1. On the one hand, by Studený (1997, Lemma 3.2), we know that for any chain graph $\mathcal{G}$, there is an edge between two vertices u and v if and only if $u \not\perp\!\!\!\perp v | S$ for any subset S of V. Thus, line 6 of Algorithm only deletes those edges that cannot appear in the true skeleton. So do lines 14 and 19.

On the other hand, if u and v are not adjacent in $\mathcal{G}$, by Theorem 3, it must be under one of the three possible conditions. If it is in condition 1, we will never connect them since we do not connect any vertex pair that is never contained in any node simultaneously. If it is in condition 2, then we will disconnect them in line 6 and line 14. Finally, if it is in condition 3, then either $u \perp\!\!\!\perp v | \mathrm{bd}_{\mathcal{G}}(u)$ or $u \perp\!\!\!\perp v | \mathrm{bd}_{\mathcal{G}}(v)$. By our previous discussion, we know that before starting line 17, we have $\mathrm{bd}_{\mathcal{G}}(u) \subset \mathrm{ne}_{\mathcal{G}'}(u)$ and $\mathrm{bd}_{\mathcal{G}}(v) \subset \mathrm{ne}_{\mathcal{G}'}(v)$ for the $\mathcal{G}'$ at that moment. Thus, we will disconnect u and v in line 19.

Correctness of Algorithm 2. By Proposition 4, Lemma 12 and the correctness of Algorithm 1, we know that every ordered vertex triple $\langle u, v, w \rangle$ in $\mathcal{G}$ with $u \to w$ a complex arrow and v the parent of (one of) the corresponding complex(es) is considered in line 4 of Algorithm 2. If the triple $\langle u, v, w \rangle$ is really in this situation, then we know that $u \not\perp\!\!\!\perp v | S_{uv} \cup \{w\}$, and hence we orient $u - w$ as $u \to w$. Moreover, Lemma 13 prevents us from orienting $u - w$ as $w \to u$ during the execution of Algorithm 2. This proves that we will orient every complex arrow right before starting line 9 in Algorithm 2.

Then by Lemma 13, the $\mathcal{G}^*$ before starting line 9 is a hybrid graph with the same pattern as $\mathcal{G}$. Thus Lemma 14 guarantees that we obtain the pattern of $\mathcal{G}$ after line 9.

Correctness of Algorithm 3. First of all, we show that any conditional independence relation represented by $\mathcal{D}$ is also represented by $\mathcal{G}$. This is straightforward by noting the following two facts:

1. assuming positivity, both DAG models and chain graph models are closed subset of graphoids under 5 axioms, see Pearl (1988) and Studený and Bouckaert (2001);

2. any conditional independence relation used in constructing $\mathcal{D}$ is represented by $\mathcal{G}$.

Then by Xie et al. (2006, Theorem 2), the $\mathcal{T}$ in line 2 is a separation tree of $\mathcal{D}$. With the block vertices substituted, by the definition of separation trees, the output $\mathcal{T}$ of Algorithm 3 is a separation tree of $\mathcal{G}$.

Correctness of Algorithm 4. Xie et al. (2006, Theorem 3) guarantees the correctness of Algorithm 4.

References

A. Agresti. *Categorical Data Analysis*. John Wiley & Sons, Hoboken, NJ., 2nd edition, 2002.

T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons, Hoboken, NJ., 3rd edition, 2003.

S. A. Andersson, D. Madigan, and M. D. Perlman. Alternative Markov properties fror chain graphs. *Scand. J. Statist.*, 28:33–85, 2001.

I. Beinlich, H. Suermondt, R. Chevaz, and G. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. In *Proceedings of the 2nd European Conference on Artificail Intelligence in Medicine*, pages 247–256. Springer-Verlag, Berlin, 1989.

S. Carroll and V. Pavlovic. Protein calassification using probabilistic chain graphs and the gene ontology structure. *Bioinformatics*, 22(15):1871–1878, 2006.

D. M. Chickering. Learning equivalence classes of bayesian-network structures. *J. Mach. Learn. Res.*, 2:445–498, 2002.

R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York, 1999.

D. R. Cox and N. Wermuth. *Multivariate Dependencies: Models, Analysis and Interpretation*. Chapman and Hall, London, 1996.

M. Drton and M. Perlman. A sinful approach to gaussian graphical model selection. *J. Stat. Plan. Infer.*, 138:1179–1200, 2008.

D. Edwards. *Introduction to Graphical Modelling*. Springer-Verlag, New York, 2nd edition, 2000.

B. Ellis and W. H. Wong. Learning bayesian network structures from experimental data. URL http://www.stanford.edu/group/wonglab/doc/EllisWong-061025.pdf. 2006.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007. doi: doi:10.1093/biostatistics/kxm045.

N. Friedman and D. Koller. Being bayesian about network structure: a bayesian approach to structure discovery in bayesian networks. *Mach. Learn.*, 50:95–126, 2003.

N. Friedman, I. Nachmana, and D. Pe'er. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificail Intelligence*, pages 206–215, Stockholm, Sweden, 1999.

M. Frydenberg. The chain graph markov property. *Scand. J. Statist.*, 17:333–353, 1990.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. Mach. Learn. Res.*, 8:616–636, 2007.

S. L. Lauritzen. *Graphical Models*. Claredon Press, Oxford, 1996.

S. L. Lauritzen and T. S. Richardson. Chain graph models and their causal interpretations (with discussion). *J. R. Statist. Soc. B*, 64:321–361, 2002.

Y. Liu, Xing E. P., and J. Carbonell. Predicting protein folds with structural repeats using a chain graph model. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34:1436–1462, 2006.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional graphical model selection using ℓ_1-regularized logistic regression. Technical Report 750, Department of Statistics, University of California, Berkeley., 2008. URL http://www.stat.berkeley.edu/tech-reports/750.pdf.

A. Roverato and L. La Rocca. On block ordering of variables in graphical modelling. *Scand. J. Statist.*, 33:65–81, 2006.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, Cambridge, MA, 2nd edition, 2000.

E. Stanghellini, K. J. McConway, and D. J. Hand. A discrete variable chain graph for applicants for credit. *J. R. Statist. Soc. C*, 48:239–251, 1999.

M. Studený. A recovery algorithm for chain graphs. *Int. J. Approx. Reasoning*, 17:265–293, 1997.

M. Studený and R. R. Bouckaert. On chain graph models for description of conditional independence structures. *Ann. Statist.*, 26:1434–1495, 2001.

I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Mach. Learn.*, 65:31–78, 2006.

N. Wermuth. On block-recursive linear regression equations. *Revista Brasileira de Probabilidade e Estatistica*, 6:1–56, 1992.

N. Wermuth and S. L. Lauritzen. Graphical and recursive models for contingency tables. *Biometrika*, 72:537–552, 1983.

N. Wermuth and S. L. Lauritzen. On substantive research hypotheses, conditional independence graphs and graphical chain models. *J. R. Statist. Soc. B*, 52:21–50, 1990.

J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons, 1990.

S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Ann. Math. Stat.*, 20:595–601, 1938.

X. Xie and Z. Geng. A recursive method for structural learning of directed acyclic graphs. *J. Mach. Learn. Res.*, 9:459–483, 2008.

X. Xie, Z. Geng, and Q. Zhao. Decomposition of structural learning about directed acyclic graphs. *Artif. Intell.*, 170:442–439, 2006.